Cartes auto-organisatrices pour la classification de données symboliques mixtes, de données de type intervalle et de données discrétisées.

Chantal Saad Hajjar

N° d'ordre : 2014-06-TH

SUPÉLEC

ÉCOLE DOCTORALE STITS
Sciences et Technologies de l'Information, des Télécommunications et des Systèmes

THÈSE DE DOCTORAT

DOMAINE STIC
Spécialité : Traitement du signal

Soutenue le 10 février 2014

par :

Chantal SAAD HAJJAR

Cartes auto-organisatrices pour la classification de données symboliques mixtes, de données de type intervalle et de données discrétisées

Sous la direction de :

Gilles FLEURY

Composition du jury :

Président	Edwin DIDAY	Professeur émérite, Université Paris Dauphine, France
Rapporteurs	Stéphane CANU	Professeur, INSA de Rouen, France
	Jean-Yves TOURNERET	Professeur, ENSEEIHT, France
Examinateurs	Lynne BILLARD	Professeur, University of Georgia, USA
	Ali MOHAMMAD-DJAFARI	Directeur de recherche, CNRS, France
Directeur de thèse	Gilles FLEURY	Professeur, Directeur de la recherche, Supélec, France
Encadrant	Hani HAMDAN	Professeur adjoint, Supélec, France

Résumé

Cette thèse s'inscrit dans le cadre de la classification automatique de données symboliques par des méthodes géométriques bio-inspirées, plus spécifiquement par les cartes auto-organisatrices. Nous mettons en place plusieurs algorithmes d'apprentissage des cartes auto-organisatrices pour classifier des données symboliques mixtes ainsi que des données de type intervalle et des données discrétisées. Plusieurs jeux de données symboliques simulées et réelles, dont deux construits dans le cadre de cette thèse, sont utilisés pour tester les méthodes proposées. En plus, nous proposons une carte auto-organisatrice pour les données discrétisées *(binned data)* dans le but d'accélérer l'apprentissage des cartes classiques et nous appliquons la méthode proposée à la segmentation d'images.

Abstract

This thesis concerns the clustering of symbolic data with bio-inspired geometric methods, more specifically with Self-Organizing Maps. We set up several learning algorithms for the self-organizing maps in order to cluster mixed-feature symbolic data as well as interval-valued data and binned data. Several simulated and real symbolic data sets, including two sets built as part of this thesis, are used to test the proposed methods. In addition, we propose a self-organizing map for binned data in order to accelerate the learning of standard maps, and we use the proposed method for image segmentation.

Remerciements

La réalisation de cette thèse a été possible grâce à la contribution de plusieurs personnes à qui j'adresse mes remerciements les plus sincères.

Tout d'abord, je tiens à remercier l'encadrant de ma thèse, Monsieur Hani HAMDAN, Professeur à Supélec, pour avoir proposé le sujet de cette thèse. Je le remercie surtout du temps qu'il a consacré à guider et orienter mes travaux de recherche, ainsi que de la confiance qu'il m'a accordée durant ces trois années.

Je remercie vivement le directeur de ma thèse, Monsieur Gilles FLEURY, Professeur et Directeur de la Recherche et des Relations Industrielles à Supélec, pour les conseils stimulants que j'ai eu l'honneur de recevoir de sa part lors de nos réunions. Je le remercie surtout pour sa relecture méticuleuse de mon manuscrit et pour les efforts déployés pour que cette thèse aboutisse dans les meilleurs délais.

J'adresse ma profonde reconnaissance au Professeur Stéphane FONT, Chef du Département de Traitement du Signal et Systèmes Électroniques de Supélec, de l'accueil chaleureux qu'il m'a réservé au sein de son département. Je tiens aussi à mentionner le plaisir que j'ai eu à travailler avec tous les membres de ce département et je les en remercie.

Un grand merci au Professeur Gilles DUC, Directeur adjoint de l'école doctorale STITS, pour avoir montré un intérêt particulier à l'avancement de ma thèse. Je le remercie surtout pour les entretiens que nous avons faits ensemble lors des réinscriptions.

J'exprime toute ma reconnaissance envers le Professeur Edwin DIDAY de m'avoir accordé l'honneur de présider le jury de ma soutenance. Je le remercie aussi des discussions scientifiques que nous avons eues ensemble et qui étaient très valorisantes pour mes travaux de recherche.

Je voudrais remercier particulièrement le Professeur Jean-Yves TOURNERET d'avoir rapporté ma thèse avec beaucoup de soin et d'avoir fourni une liste de commentaires constructifs dans le but d'améliorer mon mémoire.

Je remercie également le Professeur Stéphane CANU d'avoir accepté de rapporter ma thèse. J'ai surtout apprécié les perspectives qu'il a proposées à mes travaux de recherche.

Toute ma gratitude au Professeur Lynne BILLARD d'être venue spécialement des États-Unis pour participer au jury de ma soutenance et d'avoir suggéré des modifications enrichissantes sur le document de thèse.

Je remercie aussi le Professeur Ali MOHAMMAD DJAFARI d'avoir participé au jury de ma soutenance. Je suis reconnaissante aux remarques pertinentes qu'il a évoquées sur mes travaux de recherche.

Finalement, je remercie mes parents pour leurs encouragements et leur assistance aussi bien matérielle que morale qui m'ont permis de faire cette thèse dans de bonnes conditions. Je remercie chaudement ma mère Rosette, ma soeur Pascale et son mari Rend ainsi que ma soeur Mirose. Je remercie mon cher époux Nadim pour son soutien quotidien indéfectible et son enthousiasme contagieux à l'égard de mes travaux. Je remercie aussi bien mes enfants chéris Céline et Christian pour leur amour et leur support. Enfin, j'aurais tellement souhaité que mon père Mounir, qui nous a quittés avant l'aboutissement de cette thèse, aurait pu partager cette réussite avec nous. Que Dieu accorde paix à ton âme cher Père !

À ma mère Rosette qui a vivement contribué
à l'accomplissement de ce travail par son soutien et ses prières.

Table des matières

Introduction générale **1**

Contexte général 1

Objectif et apports de la thèse 3

Organisation de la thèse 5

1 Méthodes de classification **7**

1.1 Introduction 7

1.2 Aperçu général sur les méthodes de classification non supervisée . . . 8

1.2.1 Méthodes géométriques 8

1.2.2 Méthodes probabilistes 10

1.2.3 Autres méthodes 11

1.3 Quelques méthodes géométriques de classification 12

1.3.1 Classification hiérarchique ascendante 16

1.3.2 Centres-mobiles 19

1.3.3 Cartes auto-organisatrices 25

1.4 Évaluation d'une partition 39

1.4.1 Critères externes 39

1.4.2 Critères internes 41

1.5 Conclusion 43

2 Classification de données symboliques mixtes 45

2.1 Introduction 45

2.2 Données symboliques 46

2.2.1 Types de variable symbolique 47

2.2.2 Définition d'un objet symbolique. 48

2.2.3 Tableau de données symboliques 49

2.3 Mesures de proximité entre observations symboliques mixtes 51

2.3.1 Dissimilarité de Gowda et Diday 52

2.3.2 Distance de Minkowski généralisée pour les données symboliques mixtes 55

2.3.3 Autres mesures 59

2.4 Méthodes de partitionnement existantes pour les données symboliques 61

2.4.1 Extension de l'algorithme des centres-mobiles 62

2.4.2 Méthode des nuées dynamiques 64

2.4.3 Cartes auto-organisatrices pour les dissimilarités 65

2.4.4 Cartes auto-organisatrices pour les chaînes de caractères . . . 66

2.5 Cartes auto-organisatrices pour les données symboliques 68

2.5.1 1ère approche : Carte auto-organisatrice pour des données homogénéisées 68

2.5.2 2ème approche : Carte auto-organisatrice pour des données mixtes 74

2.6 Étude expérimentale 76

2.6.1 Jeu de données Huiles 76

2.6.2 Jeu de données Températures 81

2.7 Conclusion 87

3 Cartes auto-organisatrices pour les données de type intervalle 89

3.1 Introduction 89

3.2 Données de type intervalle et notations 90
3.3 Mesures de proximité entre deux vecteurs d'intervalles 91
3.3.1 Distance L_2 91
3.3.2 Distance L_1 92
3.3.3 Distances basées sur la distance de Hausdorff 92
3.3.4 Distance *vertex-type* 94
3.3.5 Distance basée sur la distance de Mahalanobis 94
3.4 Normalisation de données de type intervalle 95
3.4.1 Distance L_1 95
3.4.2 Carré de la distance L_2 95
3.4.3 Distance de Hausdorff-L_1 96
3.5 Méthodes de classification existantes pour les données intervalles . . . 96
3.6 Cartes auto-organisatrices pour les données de type intervalle 99
3.6.1 Apprentissage heuristique séquentiel 100
3.6.2 Apprentissage en mode différé en optimisant un critère 102
3.6.3 Apprentissage heuristique en mode différé 108
3.7 Étude expérimentale 118
3.7.1 Jeu de données simulées 119
3.7.2 Jeu de données France-météo 124
3.7.3 Jeu de données Liban-météo 133
3.7.4 Jeu de données *Car models* 139
3.8 Conclusion 143

4 Cartes auto-organisatrices pour les données discrétisées 145

4.1 Introduction 145
4.2 Données discrétisées 147
4.3 Cartes auto-organisatrices pour les données discrétisées 148

4.3.1 Apprentissage de la carte en mode différé en optimisant un critère . . . 148
4.3.2 Apprentissage heuristique de la carte en mode différé . . . 151
4.4 Étude expérimentale . . . 153
4.4.1 Jeu de données simulées . . . 154
4.4.2 Jeux de données réelles - Segmentation d'images . . . 163
4.5 Conclusion . . . 176

Conclusion générale **179**
Récapitulatif . . . 179
Avantages et limites . . . 181
Perspectives . . . 182

A Jeux de données de type intervalle **185**

B ACP pour les données de type intervalle **189**

C Minimiser une somme d'écarts absolus pondérés **191**

D Minimiser une somme d'écarts au carré pondérés **193**

Références bibliographiques **195**

Table des figures

1.1 Méthodes de classification non supervisée. 12

1.2 Formes des ensembles de points équidistants de l'origine d'une unité de distance pour les trois distances. 15

1.3 Forme des ensembles de points équidistants de l'origine d'une unité dans le cas tridimensionnel. 15

1.4 Dendrogramme. 18

1.5 Ensemble de données avec partitions emboîtées. 18

1.6 Carte auto-organisatrice rectangulaire de 63 neurones. 28

1.7 Mise à jour des vecteurs prototypes. 28

1.8 Fonction de voisinage gaussienne. 29

1.9 Carte auto-organisatrice pour le jeu « *simplecluster_dataset.mat* ». . 32

1.10 Classification en deux étapes. 37

2.1 Comparaison entre deux intervalles disjoints avec la distance de Ichino et Yaguchi. 58

2.2 Dendrogramme avec le critère du saut minimal pour le jeu de données Huiles. 77

2.3 Projection des données et des prototypes pour le jeu de données Huiles. 78

2.4 Carte auto-organisatrice obtenue pour le jeu de données Huiles avec l'algorithme *SOM* pour données homogénéisées. 79

2.5 Carte auto-organisatrice pour le jeu de données Huiles obtenue avec l'algorithme *SOM* pour données mixtes. 81

2.6 Projection des données et des prototypes pour le jeu de données Températures. 83

2.7 Carte auto-organisatrice obtenue pour le jeu de données Températures avec l'algorithme *SOM* pour données homogénéisées. 84

2.8 Répartition des classes sur la carte du monde avec l'algorithme *SOM* pour données homogénéisées. 85

2.9 Carte auto-organisatrice obtenue pour le jeu de données Températures avec l'algorithme *SOM* pour données mixtes. 86

2.10 Répartition des classes sur la carte du monde avec l'algorithme *SOM* pour données mixtes. 86

3.1 Représentation graphique d'une observation décrite par des variables de type intervalle. 91

3.2 Méthodes de partitionnement existantes pour les données de type intervalle. 99

3.3 Jeu de données simulées ponctuelles. 120

3.4 Données et carte après apprentissage pour le jeu de données simulées avec la méthode intSOM-DMahalanobis. 122

3.5 Données et carte après apprentissage pour le jeu de données simulées avec la méthode intSOM-CMahalanobis. 122

3.6 Données et carte après apprentissage pour le jeu de données simulées avec la méthode intSOM-L_2. 123

3.7 Répartition des stations françaises sur les 9 neurones avec la méthode intSOM-DMahalanobis. 125

3.8 Répartition des classes sur la carte géographique de la France avec la méthode intSOM-DMahalanobis. 126

3.9 ACP des vecteurs prototypes et des données du jeu de données France-météo avec la méthode intSOM-DMahalanobis. 127

3.10 Répartition des stations françaises sur les 9 neurones avec la méthode intSOM-CMahalanobis. 128

3.11 Répartition des classes sur la carte géographique de la France avec la méthode intSOM-CMahalanobis. 129

3.12 ACP des vecteurs prototypes et des données du jeu de données France-météo avec la méthode intSOM-CMahalanobis. 130

3.13 Répartition des stations françaises sur les 9 neurones avec la méthode intSOM-L_2. 131

3.14 Répartition des classes sur la carte géographique de la France avec la méthode intSOM-L_2. 131

3.15 ACP des vecteurs prototypes et des données du jeu de données France-météo avec la méthode intSOM-L_2. 132

3.16 Répartition des stations du jeu de données Liban-météo sur les 4 neurones avec la méthode intSOM-DYN-Hausdorff. 135

3.17 Répartition des classes du jeu de données Liban-météo sur la carte géographique du Liban avec la méthode intSOM-DYN-Hausdorff. . . 136

3.18 ACP des vecteurs prototypes et des données du jeu de données Liban-météo avec la méthode intSOM-DYN-Hausdorff. 137

3.19 Répartition des modèles de voitures sur les 4 neurones de la carte. . . 140

3.20 ACP des vecteurs prototypes et des données du jeu de données *Car models* avec la méthode intSOM-DYN-L_1. 141

4.1 Jeu de données simulées de six classes à proportions égales. 154

4.2 Résulats du regroupement en *bins* du jeu de données simulées. 155

4.3 Carte auto-organisatrice après apprentissage sur les *bins* classés en 6 classes à proportions égales avec la méthode binSOM-DYN-L_2. 156

4.4 Variation du *OERC* et des temps de discrétisation et d'apprentissage en fonction du nombre de *bins* par dimension. 158

4.5 Jeu de données simulées de six classes à proportions différentes. . . . 160

4.6 Carte auto-organisatrice après apprentissage sur les *bins* classés en 6 classes à proportions différentes. 160

4.7 Carte auto-organisatrice et classification hiérarchique pour le jeu de données simulées de 6 classes à proportions différentes. 162

4.8 Image Arbre. 165

4.9 Carte auto-organisatrice après apprentissage pour projeter les *bins* de l'image Arbre. 165

4.10 Dendrogramme des prototypes de la carte pour l'image Arbre. 166

4.11 Segmentation en 2 couleurs de l'image Arbre avec la méthode binSOM-L_2+intHC. 167

4.12 Segmentation en 2 couleurs de l'image Arbre avec la méthode des centres-mobiles. 168

4.13 Variation de l'indice $intDB$ en fonction du nombre de classes pour l'image Arbre. 169

4.14 Segmentation en 5 couleurs de l'image Arbre avec la méthode binSOM-L_2+intHC. 170

4.15 Segmentation en 5 couleurs de l'image Arbre avec la méthode des centres-mobiles. 170

4.16 Image Chaise. 172

4.17 Regroupement en *bins* de l'image Chaise. 172

4.18 Carte auto-organisatrice après apprentissage pour projeter les *bins* de l'image Chaise. 173

4.19 Variation de l'indice $intDB$ en fonction du nombre de classes pour l'image Chaise. 174

4.20 Segmentation en 3 couleurs de l'image Chaise avec la méthode binSOM-L_2+intHC. 174

4.21 Segmentation en 3 couleurs de l'image Arbre avec la méthode des centres-mobiles. 175

4.22 Segmentation en 8 couleurs de l'image Chaise avec la méthode binSOM-L_2+intHC. 175

4.23 Segmentation en 8 couleurs de l'image Arbre avec la méthode des centres-mobiles. 176

Liste des tableaux

1.1 Forme des ensembles de points équidistants de l'origine d'une unité pour les trois distances et par dimension. 15

1.2 Tableau de contingence entre deux partitions. 40

2.1 Tableau de données symboliques. 50

2.2 Tableau de données à variables univaluées. 50

2.3 Agrégation du tableau 2.2 par cours. 51

2.4 Tableau de données symboliques homogènes. 70

2.5 Tableau de données de type histogramme. 71

2.6 Jeu de données Huiles. 77

2.7 Matrice des distances pour le jeu de données Huiles. 80

2.8 Résultats de la classification pour le jeu de données Huiles avec l'algorithme *S-SOM*. 81

2.9 Jeu de données Températures. 82

2.10 Résultats de la classification pour le jeu de données Températures avec l'algorithme *S-SOM*. 87

2.11 Écart-type des latitudes par classe. 87

3.1 Matrice des observations décrites par des intervalles. 90

3.2 Évaluation de la qualité de la carte pour le jeu de données simulées. . 123

3.3 Vecteurs prototypes du jeu de données France-météo suivant les 2 premières composantes principales avec la méthode intSOM-DMahalanobis. 128

3.4 Vecteurs prototypes du jeu de données France-météo suivant les premières composantes principales avec la méthode intSOM-CMahalanobis. 130

3.5 Vecteurs prototypes du jeu de données France-météo suivant les 2 premières composantes principales avec la méthode intSOM-L_2. . . . 132

3.6 Distance géographique moyenne par classe pour le jeu de données France-météo. 133

3.7 Vecteurs prototypes du jeu de données Liban-météo suivant les 2 premières composantes principales avec la méthode intSOM-DYN-Hausdorff. 137

3.8 Résultats de la classification pour le jeu de données Liban-météo avec l'algorithme des nuées dynamiques associé à la distance de Hausdorff. 138

3.9 Moyenne et écart-type des altitudes par classe pour le jeu de données Liban-météo. 138

3.10 Partition a priori du jeu de données *Car models*. 139

3.11 Évaluation de la partition obtenue du jeu de données *Car models*. . . 142

3.12 Évaluation de la partition obtenue du jeu de données *Car models* avec observations aberrantes. 142

4.1 Coordonnées des moyennes des six distributions. 154

4.2 Résultats obtenus pour le jeu de données simulées avec six classes à proportions égales. 157

4.3 Résultats obtenus pour le jeu de données simulées avec six classes à proportions égales en changeant le nombre de *bins* par dimension. . . 158

4.4 Paramètres des six distributions gaussiennes avec des proportions différentes par classe. 159

4.5 Résultats obtenus avec une classification en deux étapes pour le jeu de données simulées avec des classes à proportions différentes. 162

4.6 Neurones associés à chacun des 30 nœuds du dendrogramme de l'image Arbre. 167

4.7 Temps d'exécution pour segmenter l'image Arbre. 171

A.1 Jeu de données France-météo. 186

A.2 Jeu de données Liban-météo. 188

Introduction générale

Contexte général

Le travail développé dans cette thèse s'inscrit dans le cadre de la classification non supervisée de données symboliques par des méthodes géométriques bio-inspirées, plus spécifiquement par les cartes auto-organisatrices de Kohonen.

Grâce aux progrès réalisés en technologies de l'information, il est désormais possible de recueillir, stocker et gérer de grandes masses de données. Par conséquent, il est difficile, quand le volume des informations est important, de décrire d'une façon succincte les différentes caractéristiques associées aux données, d'où la nécessité d'automiser l'analyse de ces données. De nombreuses méthodes d'analyse de données se sont développées en parallèle avec les progrès technologiques comme l'analyse en composantes principales, l'analyse factorielle, la régression, etc. L'analyse de données par classification est un domaine très utilisé dans l'exploration des données. Elle consiste à répartir les données en groupes homogènes ou classes, dans un but informel ou décisif. Une classe regroupe des individus qui ont des caractéristiques similaires. Dans le domaine de la classification, on distingue la classification non supervisée, et le classement, appelé aussi discrimination ou classification supervisée.

En classification supervisée, les classes sont connues au préalable. Un échantillon de données bien classées est utilisé pour l'apprentissage de l'algorithme de classification. L'analyse discriminante de Fisher, les arbres de décisions, les K plus proches voisins, le classificateur de Bayes, les perceptrons multicouches et les machines à vecteurs de support sont des exemples de méthodes de classification supervisée. En revanche, la classification automatique se déroule sans apprentissage supervisé. Partant d'un ensemble d'individus décrits par des caractéristiques ou des variables, la classification automatique permet de partitionner ces individus en

groupes sans aucune connaissance *a priori*. C'est une forme d'abstraction qui va au-delà des descriptions exactes des individus pour faire appartenir à un même groupe des individus ayant des traits particuliers qui se ressemblent.

Une grande famille de méthodes de classification automatique se fonde sur l'approche géométrique pour classifier les données. Le principe de base de cette approche est d'utiliser une mesure de proximité (distance ou dissimilarité) pour comparer deux individus, et d'affecter les individus qui sont proches au sens de cette mesure plus probablement à une même classe. La classification hiérarchique et la méthode des centres mobiles *(K-means)* sont deux méthodes bien connues de la classification automatique à approche géométrique. Ajoutons à ces méthodes, les cartes auto-organisatrices *(Self-Organizing Maps)*, qui, outre leur capacité de classification, permettent de renseigner sur la proximité des classes en préservant la topologie des données.

Une **carte auto-organisatrice** est un type de réseau de neurones artificiels constitué d'un ensemble de neurones arrangés suivant une grille à faible dimension. À chaque neurone est associé un vecteur prototype appartenant à l'espace des observations. L'apprentissage du réseau se fait de manière non supervisée, en mode incrémental ou en mode différé (en mode *batch*), en vue de projeter les observations sur la carte. Dans la version incrémentale, chaque itération du processus d'apprentissage consiste à présenter aléatoirement une observation au réseau, et à mettre à jour les vecteurs prototypes des neurones en vue de les rapprocher de cette observation. Par contre, en mode différé, toutes les observations sont présentées au réseau à chaque itération, et les vecteurs prototypes des neurones sont mis à jour. En fin d'apprentissage, soit incrémental soit en mode différé, une observation sera affectée au neurone dont le vecteur prototype en est le plus proche. La classification des données peut se faire en supposant que les observations affectées au même neurone appartiennent à la même classe. Dans ce cas, le nombre de neurones doit correspondre au nombre de classes et deux neurones voisins sur la carte représenteront des classes voisines dans l'espace des observations. Il est possible aussi de projeter les données sur une grande carte, puis de classifier les neurones de la carte pour achever la classification finale des données.

Les cartes auto-organisatrices, ainsi que les autres méthodes de classification, ont été initialement définies pour des données quantitatives univaluées. Cependant, les données issues des expériences réelles ne sont pas toujours formatées avec des

valeurs réelles univaluées, mais se présentent sous forme de données qualitatives, d'ensembles de valeurs quantitatives ou qualitatives ou bien même sous forme d'intervalles ou histogrammes. Il s'agit alors de **données symboliques**. Par exemple, pour exprimer les connaissances linguistiques d'un individu, on a recours à une variable pouvant contenir un ensemble de valeurs qualitatives. Donc, une variable symbolique existe sous plusieurs formes et contribue alors à une représentation plus concrète des données. Il est bien évident qu'en utilisant des variables de type intervalle pour enregistrer les températures minimales et maximales, nous obtiendrons une analyse plus réaliste du climat en question plutôt qu'avec des températures moyennes. Les données symboliques peuvent être issues directement des expériences. Elles peuvent résulter aussi de l'agrégation d'un grand jeu de données à valeurs univaluées, dans le but d'en réduire la taille en vue de faciliter l'analyse des données qu'il contient.

Le formalisme des données symboliques a suscité l'intérêt de plusieurs chercheurs qui ont contribué à instaurer l'analyse de données symboliques (ADS). L'ADS consiste à étendre la problématique et les méthodes classiques d'analyse de données pour prendre en compte les données symboliques. Plusieurs travaux ont été menés sur la classification automatique de données symboliques. Dans ce contexte, de nouvelles distances et dissimilarités ont été proposées permettant de comparer des observations décrites par des variables symboliques, ce qui a permis d'étendre les méthodes de classification basées sur des approches géométriques à ce type de données. Parmi ces méthodes, on trouve la classification hiérarchique et la méthode des nuées dynamiques qui ont été étendues au cas de données symboliques.

Objectif et apports de la thèse

Vu l'importance et les avantages des cartes auto-organisatrices dans la classification de données avec préservation de la topologie, nous avons choisi d'approfondir nos recherches dans ce domaine dans le but de les intégrer au sens large dans la classification de données symboliques. À cette fin, nous avons proposé plusieurs algorithmes d'apprentissage en mode différé (en mode *batch*) des cartes auto-organisatrices, à partir de plusieurs types de données symboliques.

Pour les données décrites par des variables symboliques mixtes, nous avons proposé deux méthodes. La première, constituée de deux étapes, consiste à normaliser les données en premier lieu dans le but de les transformer en données histogrammes,

et en second lieu, à utiliser ces données normalisées pour entraîner la carte auto-organisatrice en mode différé en utilisant la distance euclidienne pour comparer deux histogrammes. La deuxième méthode est basée sur un algorithme qui permet d'entraîner une carte auto-organisatrice avec des données de proximité plutôt qu'avec des individus. Ces distances sont calculées en utilisant une distance adéquate permettant de comparer deux observations décrites par des variables symboliques mixtes. Les deux méthodes ont été testées et comparées à d'autres méthodes de classification de données symboliques moyennant un jeu de données symboliques mixtes réelles et un jeu de données de type intervalle réelles.

Une grande part de notre travail est consacrée aux données de type intervalle vu leur utilisation croissante en analyse de données du fait qu'elles permettent de modéliser l'incertitude ainsi que la variabilité dans les mesures. Deux familles d'algorithmes ont été proposées pour l'apprentissage des cartes auto-organisatrices à partir de données de type intervalle :

- La première famille comprend trois méthodes où l'apprentissage se fait en optimisant un critère qui dépend d'une distance entre vecteurs d'intervalles. La première méthode utilise une extension de la distance de Minkowski de type L_2 aux données intervalles. La deuxième méthode utilise une extension de la distance de Minkowski de type L_1 aux données intervalles. La troisième méthode utilise une combinaison de type L_1 de distances de Hausdorff adaptées à la comparaison de deux intervalles.
- La deuxième famille regroupe des algorithmes où l'apprentissage se fait d'une manière heuristique mais toujours en mode différé, avec les mêmes distances utilisées dans la première famille. En plus, deux méthodes d'apprentissage adaptatif sont proposées en utilisant une extension de la distance de Mahalanobis aux données intervalles. La première méthode utilise une distance adaptative commune à toutes les classes, alors que dans la deuxième méthode, l'apprentissage commence avec une distance adaptative commune à toutes les classes et termine avec une distance adaptative différente par classe, ce qui conduit à une classification adaptative des données. Par ailleurs, ces deux méthodes produisent des résultats de classification plus satisfaisants qu'avec une distance de Minkowski de type L_2 du fait que la distance de Mahalanobis permet de prendre en compte la variabilité des variables ainsi que la corrélation entre les variables.

Pour tester les différentes méthodes proposées pour les données intervalles, nous

avons conçu deux jeux de données intervalles constitués à partir de données météorologiques françaises et libanaises. Nous avons utilisé aussi d'autres jeux de données intervalles existants en vue de comparer nos méthodes avec d'autres méthodes de classification.

Nous nous sommes intéressés aussi à la classification de données discrétisées *(binned data)*, qui sont des données qui résultent du regroupement des données ponctuelles dans des *bins* produisant ainsi un histogramme multidimensionnel. Ce sont les *bins* qui sont classifiés à la place des observations, rendant possible la classification de grands jeux de données. Notre contribution dans ce domaine s'est focalisée sur les cartes auto-organisatrices pour les données discrétisées où les vecteurs prototypes sont des vecteurs d'intervalles. Nous avons alors proposé deux algorithmes d'apprentissage d'une carte auto-organisatrice pour les données discrétisées : dans le premier algorithme, l'apprentissage se fait en mode différé, mais en optimisant un critère qui dépend de la distance L_2 qui est utilisée pour comparer deux *bins*, alors qu'il est fait d'une manière heuristique dans le deuxième algorithme. Un jeu de données simulées est construit dans le but de valider les méthodes proposées. En plus, nous avons utilisé les cartes auto-organisatrices pour les données discrétisées afin de mettre en place une technique de segmentation d'images en trois étapes : regrouper les pixels de l'image en *bins* dans une première étape, les projeter à la carte auto-organisatrice dans une deuxième étape et classifier les prototypes de la carte dans une troisième étape pour achever la classification finale des pixels.

Organisation de la thèse

La thèse est organisée en quatre chapitres et deux annexes.

Le **premier chapitre** est consacré aux méthodes de la classification automatique qui s'appliquent aux données quantitatives univaluées et représentées par des points. Nous nous intéressons plus spécifiquement aux méthodes qui utilisent les distances pour classifier les individus. Nous commençons ce chapitre par un aperçu général des méthodes de classification automatique. Ensuite, nous abordons les principes de la classification hiérarchique ascendante, de la méthode des centres mobiles et de la méthode des nuées dynamiques, vu leur importance dans le cadre de notre travail. Nous exposons en détail le principe des cartes auto-organisatrices et leur algorithmes d'apprentissage en mettant en relief leur rôle dans la classification de

données. Nous incluons à la fin de ce chapitre, les critères internes et externes les plus connus pour évaluer une partition résultante d'un algorithme de classification.

La classification des données symboliques mixtes fait l'objet du **deuxième chapitre**. Ce chapitre est divisé en deux parties. La première partie contient un état de l'art du travail réalisé dans ce domaine, comprenant les mesures de proximités existantes pour comparer deux observations décrites par des variables symboliques mixtes, ainsi que les méthodes de classification automatique les plus utilisées pour ce type de données. Dans la deuxième partie, nous proposons deux méthodes de classification pour les données symboliques moyennant les cartes auto-organisatrices. Ces méthodes sont testées et comparées à d'autres méthodes dans le cadre d'une étude expérimentale comprenant deux jeux de données symboliques réelles.

Le **troisième chapitre** est dédié à la classification automatique de données de type intervalle. Au début, nous présentons les distances entre vecteurs d'intervalles qui sont souvent utilisées dans les algorithmes de classification de ce type de données, puis nous passons en revue les méthodes existantes pour la classification des données intervalles. Au cœur de ce chapitre, nous proposons plusieurs algorithmes d'apprentissage des cartes auto-organisatrices pour les données de type intervalle. Une étude expérimentale, en utilisant un jeu de données simulées et trois jeux de données réelles, est menée dans le but de tester et de comparer les méthodes proposées entre elles-mêmes et avec d'autres méthodes récentes de classification de données intervalles.

Le **quatrième chapitre** concerne la classification des données discrétisées. Nous commençons par expliquer le formalisme des données discrétisées, pour ensuite présenter les méthodes que nous proposons pour classifier ce type de données par les cartes auto-organisatrices. Ensuite, nous testons les méthodes proposées en utilisant un jeu de données simulées. En utilisant les données discrétisées et les cartes auto-organisatrices, nous proposons une nouvelle technique de segmentation d'images dont les avantages sont mis en évidence moyennant deux images réelles.

Le **premier annexe** à la fin du document contient les deux jeux de données intervalles que nous avons construits dans le cadre de cette thèse. Le **deuxième annexe** contient une explication détaillée de l'Analyse en Composantes Principales (ACP) pour les données de type intervalle.

Chapitre 1

Méthodes de classification

1.1 Introduction

Les méthodes de classification ont été initialement développées pour analyser des données quantitatives univaluées. Ces données se présentent sous forme d'un tableau où les lignes sont les individus, appelés aussi observations, et les colonnes sont les variables. Une cellule de ce tableau contient un nombre qui représente la valeur de la variable de la même colonne que prend l'individu de la même ligne. Un individu de ce tableau est représenté graphiquement par un point de l'espace $\mathbb{R}^p$ où p est le nombre de variables ou la dimension des données.

Dans ce chapitre, nous exposons les différentes méthodes de classification non supervisée qui s'appliquent aux données numériques univaluées, en donnant plus d'importance aux méthodes à approche géométrique qui se basent sur des distances, plus spécifiquement aux méthodes qui sont utilisées dans le cadre de cette thèse.

Tout d'abord, un aperçu général sur les méthodes de classification non supervisée est présenté. Ensuite, nous exposons les mesures de proximité existantes entre individus ponctuels et détaillons la classification hiérarchique ascendante, ainsi que la méthode des centres-mobiles et ses variantes. La plus grande partie de ce chapitre est consacrée aux cartes-organisatrices de Kohonen dans leur cadre classique (opérant sur des points), plus spécifiquement à leur rôle dans la classification de données. Avant de conclure, nous parlons des différents critères existants dans la littérature pour évaluer une partition résultante d'un algorithme de classification.

1.2 Aperçu général sur les méthodes de classification non supervisée

Le but de la classification non supervisée est d'organiser les données en classes homogènes sans apprentissage supervisé, chaque classe regroupant des individus ayant des caractéristiques similaires.

Les méthodes de classification non supervisée se diversifient mais tombent essentiellement en deux catégories qui sont (voir figure 1.1, p.12) :

- Méthodes à approche géométrique.
- Méthodes à approche probabiliste.

D'autres méthodes de classification non supervisée existent aussi comme la méthode *SVC (Support Vector Clustering)* et les algorithmes génétiques.

1.2.1 Méthodes géométriques

Dans les méthodes à approche géométrique, la classification de données se fonde sur une mesure de proximité entre les individus. Deux individus qui sont proches au sens d'un certain critère de proximité appartiennent plus probablement à une même classe. Parmi ces méthodes, on trouve : la classification hiérarchique, les méthodes de classification par partitionnement, ainsi que les méthodes basées sur la densité, sur les graphes et sur les grilles.

La classification hiérarchique consiste à créer une hiérarchie sur l'ensemble des observations, produisant plusieurs partitions emboîtées dans un ordre ascendant ou descendant, en utilisant une matrice de distances ou de dissimilarités entre individus (Johnson, 1967). Quand l'ordre est ascendant, la classification se fait d'une façon agglomérative appelée **classification hiérarchique ascendante**. Par contre, quand l'ordre est descendant, il s'agit d'une **classification descendante** ou divisive du fait que toutes les observations appartiennent à une même classe au début ; ensuite, le nombre de classes croît progressivement pour qu'à la fin de l'algorithme, chaque observation constitue une classe. Cette méthode a l'avantage d'être déterministe, produisant les mêmes résultats à chaque lancée de l'algorithme, cependant, la construction d'une matrice de dissimiliarités nécessite un espace mémoire important pour les grands jeux de données.

Les méthodes de classification par **partitionnement** visent à créer une seule partition de l'ensemble de données, pouvant être **fixe** ou **floue**. Dans une partition floue, les classes se chevauchent et une observation peut appartenir à plus d'une classe, tout en quantifiant cette appartenance par un certain degré qui varie entre 0 et 1. Ce type de classification permet d'avoir des partitions plus souples tolérant des données imprécises ou incertaines. L'algorithme des c-moyennes floues (Bezdek, 1981) *(fuzzy c-means)* est un algorithme bien connu qui permet de fournir une partition floue d'un ensemble de données, en minimisant itérativement une fonction coût. Les classes sont représentées par leurs centres de gravité qui sont recalculés à chaque itération, ainsi que les degrés d'appartenance des observations aux classes jusqu'à atteindre la convergence. Par contre, avec une partition fixe, chaque observation appartient à une et une seule classe. L'algorithme des **centres-mobiles** (Forgy, 1965; MacQueen, 1967), appelé aussi *K-means*, conduit à une partition fixe de l'ensemble de données en optimisant une fonction coût, et chaque individu sera affecté à la classe dont le centre de gravité en est le plus proche. Avec cette méthode, la partition finale dépend du choix des centres initiaux. La méthode des **nuées dynamiques** (Diday, 1971) est une généralisation de la méthode des centres-mobiles, en supposant que le représentant d'une classe ne se limite plus à son centre de gravité mais peut se présenter sous n'importe quelle forme (individu, histogramme, vecteur d'intervalles, loi de probabilité, etc.). Dans l'algorithme des K-**Médoïdes** (Kaufman et Rousseeuw, 1987), le représentant d'une classe est un individu de cette classe appelé médoïde. Les médoïdes sont attribués aléatoirement au début de l'algorithme, puis, chaque médoïde est échangé avec un individu de sa classe, jusqu'à trouver celui dont la distance de tous les autres individus est minimale et qui sera alors le médoïde final. Cet algorithme a l'avantage d'utiliser n'importe quelle distance du fait de l'absence de calcul de centre de gravité ; de plus, il est plus robuste face aux points aberrants que celui des centres-mobiles, mais, sa complexité algorithmique est plus importante. La classification de données avec les **cartes auto-organisatrices** (Kohonen, 1984) mène aussi à une partition fixe où chaque observation appartient à la classe du neurone qui lui est le plus proche, tout en préservant la topologie des données, fournissant ainsi une information supplémentaire sur la proximité des classes. Dans les méthodes de classification par partitionnement, le nombre de classes doit être fourni à l'avance par le programme et les classes produites sont principalement convexes dont la forme exacte dépend de la distance utilisée.

Les méthodes de classification à base de **densité** sont aussi des méthodes de clas-

sification non supervisées où les classes sont considérées comme des zones de haute densité, séparées par des zones de basse densité formées par des objets appelés bruits ou objets frontaliers. L'algorithme le plus populaire de ces méthodes est le *DBSCAN* (Ester *et al.*, 1996). L'idée de base de cet algorithme est de choisir une observation quelconque et de déterminer le nombre d'observations qui sont situées à une distance inférieure ou égale à un certain rayon ϵ. Si le nombre d'observations est inférieur à la valeur $Minpts$, l'observation choisie est considérée un bruit, sinon une classe est créée regroupant tous ces points. Le nombre de classes n'est pas requis par le programme, ce qui constitue un avantage majeur de cette méthode par rapport aux autres ; de plus, les classes reconnues ont des formes arbitraires qui ne sont pas nécessairement convexes. Par contre, cette méthode présente des difficultés à identifier deux classes assez proches et les résultats produits sont sensibles aux paramètres $Minpts$ et ϵ.

Les méthodes à base de **grilles** ont un principe similaire aux méthodes à base de densité. Elles consistent à diviser chaque dimension de l'espace des observations en intervalles de longueur égale formant ainsi une grille composée d'unités qui ne se chevauchent pas. Une unité est considérée comme dense si le nombre d'observations dépassent un certain seuil (paramètre similaire à $Minpts$), et une classe sera formée de plusieurs unités denses connectées ensemble (Wang *et al.*, 1997).

Les méthodes à base de **graphes** se fondent sur la théorie des graphes pour classifier les données (Augustson et Minker, 1970; Zahn, 1971). Une des méthodes de ce groupe consiste à créer un arbre valué à partir d'une matrice de dissimilarités entre les individus. Les sommets de l'arbre sont les individus, et une arête de l'arbre est pondérée par la distance entre les deux individus qui la délimitent. À chaque itération, l'arête de l'arbre de longueur maximale est supprimée afin de créer une hiérarchie de partitions sur l'ensemble des observations (Elghazel, 2007a). Cette méthode ne requiert pas le nombre de classes à l'avance et permet de reconnaître des classes pas nécessairement convexes.

1.2.2 Méthodes probabilistes

Dans la famille des méthodes à **approche probabiliste**, on considère que les données sont un échantillon indépendamment tiré d'un **modèle de mélange** de plusieurs distributions et que les classes sont identifiées en supposant qu'elles regroupent les observations appartenant le plus probablement à la même distribu-

tion. En général, la loi de probabilité utilisée est la loi normale et les distributions sont alors des gaussiennes. Le but du programme de classification est de pouvoir estimer les paramètres de ces gaussiennes par maximum de vraisemblance, ce qui peut être réalisé avec l'algorithme EM (Éspeŕance/Maximisation) (Dempster *et al.*, 1977). Au début, les paramètres des distributions gaussiennes sont estimés au hasard, l'algorithme EM cherche à optimiser ces paramètres itérativement de façon à faire correspondre au mieux les distributions aux données. L'avantage de cette méthode réside dans le fait qu'elle permet la reconnaissance de classes elliptiques et bien mélangées. Cependant, la partition finale dépend du choix des probabilités initiales, et le nombre de classes doit être fourni à l'algorithme dont la convergence risque d'être lente pour les grands jeux de données.

1.2.3 Autres méthodes

D'autres méthodes de classification non supervisée existent aussi comme les méthodes à base d'**algorithmes génétiques** (Raghavan et Birchard, 1979), et la méthode à base de machines à vecteurs supports non supervisée connue sous le nom de ***SVC*** (*Support Vector Clustering*) qui consiste à projeter les données dans un espace à dimension supérieure, appelé espace de redescription, en utilisant une fonction noyau. Dans cet espace, la plus petite sphère qui englobe les données est déterminée en utilisant l'algorithme *SVDD : Support Vector Domain Description*. Cette sphère est de nouveau projetée dans l'espace initial des données, produisant une série de contours. Les points entourés par le même contour sont supposés appartenir à la même classe. De même, avec cette méthode le nombre de classes est détecté automatiquement et il n'y pas de limitation sur la forme des classes, mais, cette méthode n'est pas recommandée pour les données à haute dimensionnalité (Ben-Hur *et al.*, 2001).

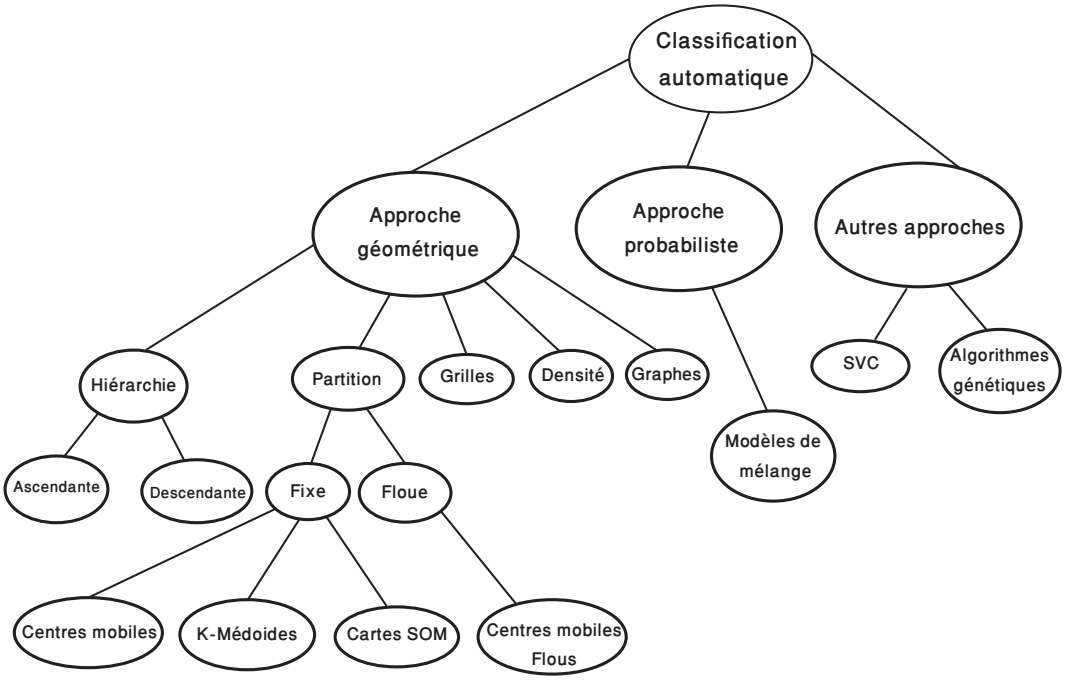


FIGURE 1.1 – Méthodes de classification non supervisée.

1.3 Quelques méthodes géométriques de classification

Nous consacrons le reste du chapitre pour exposer en détail quelques méthodes de classification non supervisée qui sont impliquées dans les travaux de recherche effectués dans cette thèse, comme la classification hiérarchique ascendante, la méthode des centres-mobiles et ses dérivés et les cartes auto-organisatrices de Kohonen. Puisque toutes ces méthodes utilisent les distances entre les individus pour pouvoir les classifier, nous commençons par citer les mesures de proximité existantes pour comparer deux observations ponctuelles. Dans tout ce chapitre, l'ensemble des n individus à classifier sera noté par Ω. Un individu est décrit par p variables quantitatives et représenté par un vecteur $\boldsymbol{x}_i \in I\!R^p$. Ce jeu de données est alors représenté par une matrice de n lignes et p colonnes notée : $\boldsymbol{\mathcal{X}} = \left(x_i^j\right)_{\substack{i\in\{1,\ldots,n\}\\ j\in\{1,\ldots,p\}}}$

Dans les méthodes à approche géométrique, classifier l'ensemble Ω consiste à faire appartenir à une même classe des individus qui se ressemblent. À cette fin,

un opérateur de comparaison est utilisé. Il peut se présenter sous forme d'une similarité pour mesurer la ressemblance entre les individus, ou bien sous forme d'une dissimilairité ou d'une distance pour mesurer la dissemblance entre les individus.

Définition 1.1. *Un opérateur de comparaison* $s : \Omega \times \Omega \to [0,1]$ *est une similarité s'il vérifie les propriétés suivantes :*

1. $s(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) \geqslant 0$ *(positivité)*
2. $s(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = s(\boldsymbol{x}_{i\prime}, \boldsymbol{x}_i)$ *(symétrie)*
3. $s(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = 1 \Leftrightarrow \boldsymbol{x}_i = \boldsymbol{x}_{i\prime}$ *(normalisation)*

Définition 1.2. *Un opérateur de comparaison* $d : \Omega \times \Omega \to I\!R^+$ *est une dissimilarité s'il vérifie les propriétés suivantes :*

1. $d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = d(\boldsymbol{x}_{i\prime}, \boldsymbol{x}_i)$ *(symétrie)*
2. $d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) \geq d(\boldsymbol{x}_i, \boldsymbol{x}_i) = 0$ *(positivité)*

Définition 1.3. *Un opérateur de comparaison* $d : \Omega \times \Omega \to I\!R^+$ *est une distance s'il vérifie les propriétés suivantes :*

1. $d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = d(\boldsymbol{x}_{i\prime}, \boldsymbol{x}_i)$ *(symétrie)*
2. $d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) \geqslant 0$ *et* $d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = 0 \Leftrightarrow \boldsymbol{x}_i = \boldsymbol{x}_{i\prime}$ *(positivité)*
3. $d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime\prime}) \leqslant d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) + d(\boldsymbol{x}_{i\prime}, \boldsymbol{x}_{i\prime\prime})$ *(inégalité triangulaire)*

Par la suite, nous allons présenter les distances utilisées pour comparer deux observations décrites par des variables quantitatives univaluées.

Définition 1.4. *On définit la distance de Minkowski entre les deux vecteurs d'observations* $\boldsymbol{x}_i$ *et* $\boldsymbol{x}_{i\prime}$ *par :*

$$d_q(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = \sqrt[1/q]{\sum_{j=1}^{p} \left| x_i^j - x_{i\prime}^j \right|^q} \tag{1.1}$$

Cette distance se réduit à la distance euclidienne pour $q = 2$, et à la distance city-block pour $q = 1$.

Définition 1.5. *On définit la distance euclidienne entre les deux vecteurs d'observations* $\boldsymbol{x}_i$ *et* $\boldsymbol{x}_{i\prime}$ *par :*

$$d_2(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = \sqrt{\sum_{j=1}^{p} \left(x_i^j - x_{i\prime}^j \right)^2} \tag{1.2}$$

Le carré de la distance euclidienne est souvent utilisé en analyse de données pour comparer deux observations. Mais, dans ce cas, il s'agit d'une dissimilarité plutôt que d'une distance car la propriété de l'inégalité triangulaire ne peut pas être vérifiée.

Définition 1.6. *On définit la distance city-block appelée aussi distance de Manhattan entre les deux vecteurs d'observations $\boldsymbol{x}_i$ et $\boldsymbol{x}_{i\prime}$ par :*

$$d_1(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = \sum_{j=1}^{p} \left| x_i^j - x_{i\prime}^j \right| \tag{1.3}$$

Et quand $q \to \infty$ on obtient la distance de Chebyshev :

$$\lim_{q\to\infty} \sqrt[1/q]{\sum_{j=1}^{p} \left| x_i^j - x_{i\prime}^j \right|^q} = \max_{j=1}^{p} \left| x_i^j - x_{i\prime}^j \right| \tag{1.4}$$

En classification de données, le choix d'une distance a une influence sur la forme des classes obtenues. Par exemple, la distance euclidienne contribue à la formation de classes hypersphériques, alors qu'avec la distance de Chebychev, les classes obtenues sont hypercubiques.

Les figures 1.2 et 1.3 montrent la forme des ensembles de points équidistants d'une unité de l'origine pour les trois distances dans le cas bidimensionnel et dans le cas tridimensionnel respectivement avec les détails dans le tableau 1.1.

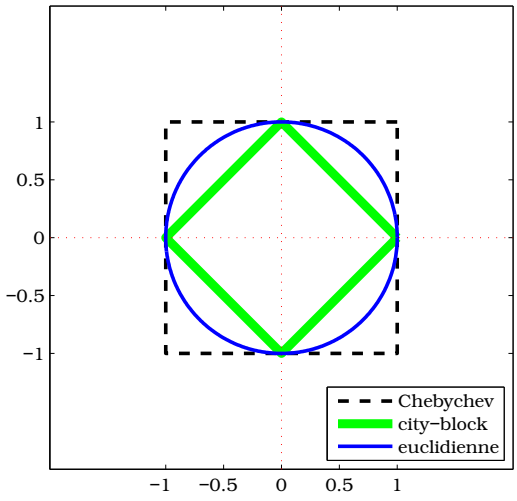
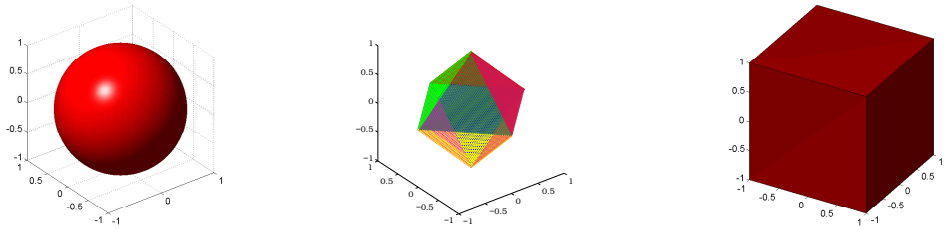


FIGURE 1.2 – Formes des ensembles de points équidistants de l'origine d'une unité de distance pour les trois distances.



(a) euclidienne (b) city-block (c) Chebychev

FIGURE 1.3 – Forme des ensembles de points équidistants de l'origine d'une unité dans le cas tridimensionnel.

Tableau 1.1 – Forme des ensembles de points équidistants de l'origine d'une unité pour les trois distances et par dimension.

Distance	**2D**	**3D**
Distance euclidienne	Cercle de centre 0 et de rayon 1	Sphère de centre 0 et de rayon 1
Distance city-block	Carré de côté $\sqrt{2}$ avec les sommets sur les axes	Octaèdre régulier de côté $\sqrt{2}$
Distance de Chebychev	Carré de côté 1 avec les côtés parallèles aux axes	Cube de côté 1

Une autre distance bien utilisée dans la classification de données est la distance

de Mahalanobis. Elle permet de prendre en compte la variabilité des variables et la corrélation entre les variables. Les classes produites suite à l'utilisation de cette distance sont plutôt ellipsoïdales.

Définition 1.7. *On définit la distance de Mahalanobis entre les deux vecteurs d'observations $\boldsymbol{x}_i$ et $\boldsymbol{x}_{i\prime}$ par :*

$$d_M(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_{i\prime})^T \boldsymbol{S}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_{i\prime})} \tag{1.5}$$

où $\boldsymbol{S}$ est la matrice de covariance intra-classe.

L'utilisation de cette distance peut causer des problèmes quant à l'inversion de la matrice de covariance. Au cas où les variables ne sont pas corrélées, la matrice de covariance devient égale à la matrice identité et la distance de Mahalanobis se réduit à la distance euclidienne.

1.3.1 Classification hiérarchique ascendante

La classification hiérarchique consiste à créer une hiérarchie de partitions sur l'ensemble des observations dans un ordre ascendant. Au début de l'algorithme, chaque individu constitue sa propre classe pour partir alors avec une partition de n classes. Ensuite, à chaque étape, on fusionne les deux classes qui se ressemblent le plus au sens d'un critère d'agrégation défini au préalable. L'algorithme s'arrête quand il n'y plus qu'une seule classe emboîtant toutes les autres ou quand le nombre de classes désiré est atteint. La dissemblance entre les individus est déterminée en calculant la distance entre leurs vecteurs respectifs, et une matrice de distances entre individus est construite à cet effet. La distance entre deux classes contenant plus d'un individu est déterminée suivant l'un des critères d'agrégation explicités ci-dessous :

- **Critère du saut minimal (Single linkage clustering)** : c'est la plus petite distance séparant un individu de la première classe et un individu de la deuxième classe :

$$D(C_1, C_2) = \min_{\boldsymbol{x}_i \in C_1, \boldsymbol{x}_{i\prime} \in C_2} d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) \tag{1.6}$$

- **Critère du saut maximal (Complete linkage clustering)** : c'est la plus grande distance séparant un individu de la première classe et un individu de la deuxième classe :

$$D(C_1, C_2) = \max_{\boldsymbol{x}_i \in C_1, \boldsymbol{x}_{i\prime} \in C_2} d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) \tag{1.7}$$

– **Critère de la moyenne (Average linkage clustering)** : c'est la moyenne des distances entre tous les individus de la première classe et tous les individus de la deuxième classe :

$$D(C_1, C_2) = \frac{1}{\big|C_1\big|\big|C_2\big|} \sum_{\boldsymbol{x}_i \in C_1} \sum_{\boldsymbol{x}_{i\prime} \in C_2} d(\boldsymbol{x}_i, \boldsymbol{x}_{i\prime}) \tag{1.8}$$

où $\big|C\big|$ désigne l'effectif de la classe C.

– **Critère de Ward** (Ward, 1963) : ce critère impose l'utilisation du carré de la distance euclidienne.

$$D(C_1, C_2) = \frac{\big|C_1\big|\big|C_2\big|}{\big|C_1\big| + \big|C_2\big|} d_2^2(G_1, G_2) \tag{1.9}$$

où $\big|C\big|$ désigne l'effectif de la classe C, G_1 le centre de gravité de la première classe et G_2 le centre de gravité de la deuxième classe.

Le choix du critère d'agrégation a une grande influence sur le résultat de la classification. Le critère du saut minimal contribue à l'obtention de classes allongées tandis que le critère du saut maximal est sensible aux points aberrants. Donc, les deux critères peuvent conduire à des formes indésirables pour les classes. Le critère de la moyenne aide à éviter les écueils des deux critères extrêmes et contribue alors à la formation de classes plus ou moins compactes. Le critère de Ward cherche, à chaque agrégation, à maximiser l'inertie inter-classe et à minimiser l'inertie intra-classe. Ce critère est souvent le plus utilisé quand le carré de la distance euclidienne est utilisé comme mesure de proximité.

Il est possible de schématiser les étapes de la classification en utilisant un dendrogramme : graphique en arbre où l'axe horizontal contient les indices des individus à classifier. Chaque deux classes fusionnées sont liées ensemble en forme de U renversé dont la hauteur exprime la distance entre les deux classes. La figure 1.4 montre le dendrogramme résultant de la classification hiérarchique ascendante, en utilisant le critère du saut minimal, d'un ensemble de données dont les individus sont tracés sur la figure 1.5.

L'algorithme 1.1 décrit toutes les étapes de cette méthode sous forme de pseudo-code. Dans le cas général, la complexité d'un tel algorithme est $O(n^3)$, ce qui le rend lent pour les grands jeux de données. Toutefois, des méthodes plus optimi-

sées de la classification agglomérative sont proposées avec une complexité de $O(n^2)$ (Sibson, 1973).

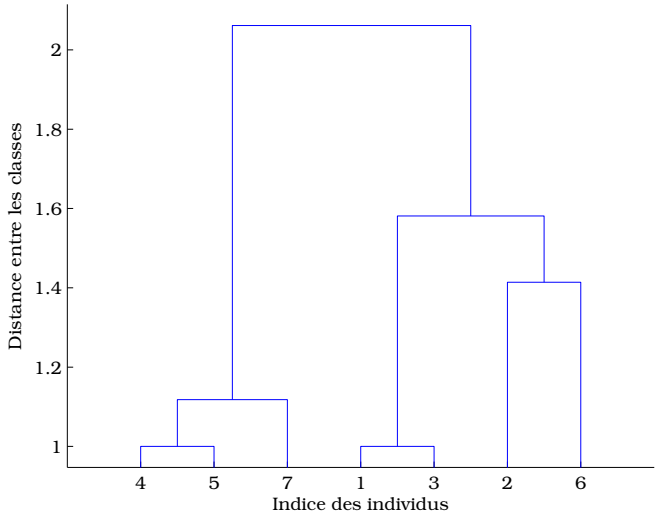


FIGURE 1.4 – Dendrogramme.

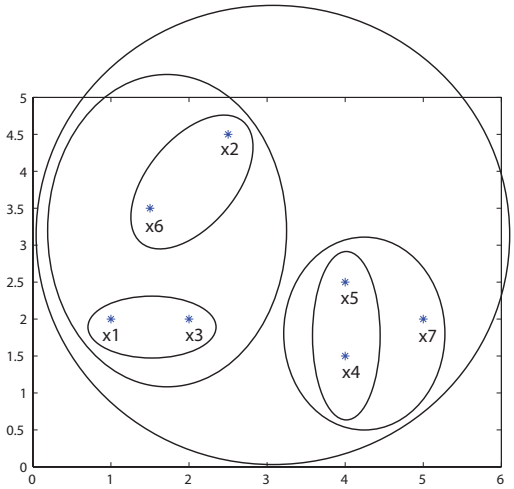


FIGURE 1.5 – Ensemble de données avec partitions emboîtées.

Algorithme 1.1 Classification hiérarchique ascendante.

Entrées : Fournir :

– $\boldsymbol{\mathcal{X}}$ {□ La matrice des individus}

– K {□ Le nombre de classes désiré, sinon K vaut 1}

Sorties : Récupérer :

– P {□ Les partitions emboîtées}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

$nbClasses \leftarrow n$ {□ Le nombre de classes est égal à celui des individus}

Mettre chaque individu dans une classe

Pour $i = 1 \rightarrow n$ **Faire**

$C_i \leftarrow \boldsymbol{x}_i$ {□ Mettre chaque individu dans une classe}

Fin pour

$P[t] = P[0] \leftarrow \{C_1, \ldots, C_n\}$ {□ Partition initiale}

Agrégations :

Tantque $nbClasses > K$ **Faire**

$t \leftarrow t + 1$

Calculer les distances entre les classes : $D(C_k, C_l)_{\substack{k \in \{1,\ldots,nbClasses\} \\ l \in \{1,\ldots,nbClasses\}}}$

Trouver les deux classes les plus proches (C_q et C_o) au sens d'un critère d'agrégation : $D(C_q, C_o) \leftarrow \min_{\substack{k \in \{1,\ldots,nbClasses\} \\ l \in \{1,\ldots,nbClasses\}}} D(C_k, C_l)$

Copier les individus de la classe C_o dans la classe C_q :

$C_q \leftarrow \boldsymbol{x}_i, \forall \boldsymbol{x}_i \in C_o$

Supprimer la classe C_o

$nbClasses \leftarrow nbClasses - 1$

$P[t] \leftarrow P[t-1] - \{C_o\}$ {□ Partition générée à l'itération t}

Fin tantque

Retourner P {□ Les partitions emboîtées}

1.3.2 Centres-mobiles

Contrairement aux méthodes hiérarchiques qui produisent différentes partitions emboîtées, les algorithmes de classification par partition répartissent les individus en classes dans le but d'obtenir une seule partition optimale. Une partition fixe sur l'ensemble Ω est constituée de K classes tel que chaque individu appartienne à une seule classe. Le nombre des classes doit être fourni à l'algorithme.

Définition 1.8. *On définit une partition fixe P de Ω avec $P = \{C_1, \ldots, C_K\}$, un*

ensemble de K parties ou classes vérifiant :

$$\begin{aligned} \forall\, k \in \{1,\ldots,K\},\ C_k \neq \phi \\ \forall\, k,l \in \{1,\ldots,K\},\ k \neq l \rightarrow C_k \neq C_l \\ \bigcup_{k\in\{1,\ldots,K\}} C_k = P \end{aligned} \tag{1.10}$$

Étant donné n individus à partitionner en K classes, le nombre de partitions possibles est :

$$N_P = \frac{1}{K!}\sum_{k=0}^{K}(-1)^{K-k}\mathrm{C}_K^k k^n \tag{1.11}$$

Parmi les algorithmes de classification par partition existent ceux qui cherchent la meilleure partition entre toutes les partitions possibles au sens d'un critère donné. Toutefois, ces méthodes deviennent très coûteuses, voire même impossibles du fait que le nombre de partitions possibles croît rapidement avec n et K. Par exemple, avec $n = 100$ individus répartis en $K = 4$ classes, nous obtenons $N_P = 6.696 \times 10^{58}$ partitions. Pour cette raison, on a recours à des algorithmes plus simples permettant de classifier de grands jeux de données en partant d'une partition initiale quelconque et en optimisant un certain critère. Nous présentons par la suite l'algorithme des centres-mobiles *(K-means)* (Forgy, 1965; MacQueen, 1967), qui est un algorithme bien connu parmi les algorithmes de classification par partition.

Le but de la méthode des centres-mobiles est de parvenir, en un nombre d'itérations limité, à partitionner les n individus en K classes homogènes en minimisant le critère suivant :

$$G_{cm} = \sum_{k=1}^{K}\sum_{i\in C_k} d_2^2(\boldsymbol{x}_i, \boldsymbol{w}_k) \tag{1.12}$$

où $\boldsymbol{w}_k$ est le représentant de la classe C_k qui est aussi son centre de gravité défini dans l'équation (1.13) et d_2 est la distance euclidienne (voir définition 1.5, p.13),

$$\boldsymbol{w}_k = \frac{\sum_{\boldsymbol{x}_i\in C_k}\boldsymbol{x}_i}{|C_k|} \tag{1.13}$$

$|C_k|$ étant l'effectif de la classe k.

Le critère G_{cm} est tout simplement la somme des inerties de toutes les classes

appelée inertie intra-classe. L'inertie totale de l'ensemble Ω étant donnée par :

$$I(\Omega) = \underbrace{\sum_{k=1}^{K} |C_k| d_2^2(\boldsymbol{w_k}, \overline{\boldsymbol{w}})}_{I_{inter}} + \underbrace{\sum_{k=1}^{K} \sum_{i \in C_k} d_2^2(\boldsymbol{x}_i, \boldsymbol{w}_k)}_{I_{intra}} \tag{1.14}$$

où $\overline{\boldsymbol{w}}$ est le centre de gravité de l'ensemble des observations.

Minimiser l'inertie intra-classe revient donc à maximiser l'inertie inter-classe du fait que l'inertie totale est constante, ce qui contribue à l'obtention de classes homogènes et bien séparées.

Partant d'une partition initiale quelconque, chaque itération de l'algorithme des centres-mobiles se résume en deux étapes principales :

- Étape représentation : calculer les centres des classes $\boldsymbol{w}_k, k \in \{1, \ldots, K\}$.
- Étape affectation : affecter les individus à la classe la plus proche (une nouvelle partition est obtenue).

L'algorithme s'arrête quand la partition ne change plus.

La distance utilisée par défaut est le carré de la distance euclidienne. Quand c'est le cas, les représentants des classes sont leurs centres de gravité (moyennes des observations) et les classes obtenues sont plutôt sphériques. Il est possible d'utiliser aussi la distance city-block d_1 (voir définition 1.6, p.14), mais, dans ce cas, le représentant d'une classe est la médiane des observations qu'elle contient et les classes obtenues sont plutôt carrées dans le cas bidimensionnel (voir figure 1.2, p.15) et en forme d'octaèdre dans le cas tridimensionnel (voir figure 1.3, p.15).

L'algorithme 1.2 détaille les étapes de la méthode des centres-mobiles avec comme mesure de proximité le carré de la distance euclidienne. La complexité de cet algorithme est linéaire en n et vaut $O(T.n.K.p)$ où T est le nombre d'itérations nécessaires pour atteindre la convergence. Donc, cet algorithme s'applique bien aux grands jeux de données.

Cependant, l'algorithme des centres-mobiles présente un inconvénient majeur du fait que la convergence vers un optimum global n'est pas garantie. En effet, un mauvais choix de la partition initiale peut causer une convergence vers un optimum local et conduire alors à une partition finale inadéquate. Il n'existe pas théoriquement des méthodes permettant de choisir judicieusement la partition initiale. Pourtant, l'une des solutions à ce problème consiste à exécuter plusieurs lancées de cet algorithme, avec à chaque fois une partition initiale différente, et la partition finale

produite par la lancée qui conduit à minimiser le plus le critère G_{cm} sera adoptée.

Algorithme 1.2 Centres-mobiles.

Entrées : Fournir :
- $\boldsymbol{\mathcal{X}}$ {□ La matrice des individus}
- K {□ Le nombre de classes désiré}

Sorties : Récupérer :
- $P_f \leftarrow \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ t : itération courante}

Choix aléatoire de la partition initiale :

Pour $i = 1 \rightarrow n$ **Faire**

$l \leftarrow alea(1, \ldots, K)$ {□ Choisir une valeur aléatoire entre 1 et K}

$C_l \leftarrow \boldsymbol{x}_i$

Fin pour

Partionnement :

Répéter

$t \leftarrow t + 1$

Étape représentation : *calculer le centre de chaque classe C_k d'effectif $|C_k|$*

Pour $k = 1 \rightarrow K$ **Faire**

$\boldsymbol{w}_k \leftarrow \frac{\sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i}{|C_k|}$

Fin pour

Étape affectation : *générer une nouvelle partition*

Pour $i = 1 \rightarrow n$ **Faire**

$d(\boldsymbol{w}_l, \boldsymbol{x}_i) \leftarrow \min_{k \in \{1,\ldots,K\}} d(\boldsymbol{w}_k, \boldsymbol{x}_i)$ {□ Trouver la classe la plus proche pour le vecteur de données courant}

$C_l \leftarrow \boldsymbol{x}_i$

Fin pour

Jusqu'à $P_t \neq P_{t-1}$

Retourner $P_f = P_t$

Centres-mobiles séquentiel

Il se peut que les observations ne soient pas disponibles toutes en même temps pour les classifier en mode différé comme c'est le cas avec la version classique des

centres-mobiles. La version séquentielle de cet algorithme permet de classifier les observations une à la fois (MacQueen, 1967). Elle consiste à déterminer la classe la plus proche pour l'observation courante et le centre de gravité de cette classe est recalculé pour prendre en compte cette nouvelle appartenance. L'algorithme 1.3 expose les étapes de cette version séquentielle des centres-mobiles.

Algorithme 1.3 Centres-mobiles séquentiel.

Entrées : Fournir :

- $\boldsymbol{\mathcal{X}}$ {□ La matrice des individus}
- K {□ Le nombre de classes désiré}

Sorties : Récupérer :

- $P_f \leftarrow \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ Itération 0}

Initialiser les représentants des classes $\{\boldsymbol{w}_k, k \in \{1, \ldots, K\}\}$ avec des valeurs aléatoires

Initialiser les effectifs des classes à 0 : $|C_1| = |C_2| = \cdots = |C_K| \leftarrow 0$

Partionnement :

Répéter

$t \leftarrow t + 1$ {Itération courante}

Étape affectation : *acquérir une observation* $\boldsymbol{x}_i$ *et l'affecter à la classe la plus proche :*

Pour $k = 1 \rightarrow K$ **Faire**

$d_2^2(\boldsymbol{w}_l, \boldsymbol{x}_i) \leftarrow \min_{k \in \{1,\ldots,K\}} d_2^2(\boldsymbol{w}_k, \boldsymbol{x}_i)$

$C_l \leftarrow \boldsymbol{x}_i$

Fin pour

Étape représentation : *calcul du nouveau centre de gravité de la classe* C_l

$n_l \leftarrow n_l + 1$

$\boldsymbol{w}_l \leftarrow \boldsymbol{w}_l + \frac{\boldsymbol{x}_i - \boldsymbol{w}_l}{n_l}$

Jusqu'à Plus d'observations disponibles

Retourner $P_f = P_t$

Nuées dynamiques

L'algorithme des nuées dynamiques est une généralisation de l'algorithme des centres-mobiles (Diday, 1971). Dans cet algorithme, le représentant ou prototype d'une classe serait plus complexe que la moyenne des observations pour être de nature quelconque : un individu de cette classe, un groupe d'individus, une loi de probabilité, une droite, un histogramme, etc. En désignant par $\boldsymbol{\gamma}_k$ le représentant de la classe k, l'algorithme des nuées dynamiques cherche à trouver, pour un ensemble d'observations Ω, la partition optimale $P = \{C_1, \ldots, C_K\}$, et ce, en partant d'une partition initiale quelconque et en minimisant d'une façon itérative le critère suivant :

$$G_{nd} = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in C_k} d(\boldsymbol{x}_i, \boldsymbol{\gamma}_k) \tag{1.15}$$

où $d : \Omega \times \Gamma \longrightarrow \mathbb{R}^+$ est une distance utilisée pour mesurer la proximité entre les vecteurs de données appartenant à l'ensemble Ω, et les vecteurs prototypes appartenant à l'ensemble Γ. Minimiser le critère G_{nd} revient à le minimiser pour chaque classe k. Le prototype $\boldsymbol{\gamma}_k$ est alors calculé en minimisant le critère g_k :

$$g_k = \sum_{\boldsymbol{x}_i \in C_k} d(\boldsymbol{x}_i, \boldsymbol{\gamma}_k) \tag{1.16}$$

L'algorithme 1.4 détaille les étapes de la méthode des nuées dynamiques. Quand la distance d est remplacée par la distance d_2^2 (carré de la distance euclidienne) et quand les prototypes des classes sont leurs centres de gravités, on retombe sur l'algorithme des centres-mobiles.

Algorithme 1.4 Nuées dynamiques.

Entrées : Fournir :
- $\boldsymbol{\mathcal{X}}$ {□ La matrice des individus}
- K {□ Le nombre de classes désiré}

Sorties : Récupérer :
- $P_f \leftarrow \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Pour $i = 1 \rightarrow n$ **Faire**

$l \leftarrow alea(1, \ldots, K)$ {□ Choisir une valeur aléatoire entre 1 et K}

$C_l \leftarrow \boldsymbol{x}_i$ {□ Partition initiale aléatoire}

Fin pour
Partionnement :
Répéter
$t \leftarrow t + 1$ {□ Itération courante}
Étape représentation : *calculer le prototype de chaque classe* C_k
Pour $k = 1 \rightarrow K$ **Faire**
calculer $\boldsymbol{\gamma}_k$ qui minimise $\sum_{\boldsymbol{x}_i \in C_k} d(\boldsymbol{x}_i, \boldsymbol{\gamma}_k)$
Fin pour
Étape affectation : *générer une nouvelle partition*
Pour $i = 1 \rightarrow n$ **Faire**
$d(\boldsymbol{\gamma}_l, \boldsymbol{x}_i) \leftarrow \min_{k \in \{1,\ldots,K\}} d(\boldsymbol{\gamma}_k, \boldsymbol{x}_i)$ {□ Trouver la classe la plus proche pour le vecteur de données courant}
$C_l \leftarrow \boldsymbol{x}_i$
Fin pour
Jusqu'à $P_t \neq P_{t-1}$
Retourner $P_f = P_t$

1.3.3 Cartes auto-organisatrices

Les cartes auto-organisatrices *(Self-Organizing Maps (SOM))*, furent inventées par Kohonen (1984). Inspirée par le principe neuronal du cerveau des mammifères, une carte auto-organisatrice est un type de réseau de neurones artificiels dont l'apprentissage se déroule de manière non supervisée. Leur rôle principal est de faire une projection non linéaire des données de haute dimension sur un espace de faible dimension. Les cartes auto-organisatrices sont largement utilisées dans la classification de données.

Dans ce qui suit, nous allons expliquer le principe des cartes auto-organisatrices en détaillant leurs algorithmes d'apprentissage. Nous allons parler aussi de l'évaluation de la performance de la carte, du choix de ses paramètres d'apprentissage, de ses critères de convergence, de son rôle dans la classification de données. Enfin, nous allons évoquer les différentes variantes qui en émergent.

Composition et principe

Une carte auto-organisatrice est composée d'une grille de neurones de faible dimension. Quand la grille est unidimensionnelle, chaque neurone a deux voisins. Quand la grille est bidimensionnelle, l'arrangement des neurones se fait d'une façon rectangulaire où chaque neurone possède 4 voisins (topologie rectangulaire) ou d'une façon hexagonale où chaque neurone possède 6 voisins (topologie hexagonale).

Les neurones sont reconnus par leur numéro et leur emplacement sur la grille. Par exemple, la figure 1.6 représente une carte rectangulaire de 7 lignes et 9 colonnes. L'emplacement du neurone 10 est désigné par le vecteur $\boldsymbol{r}_{10} = (3, 2)^T$.

Les données sont projetées de leur espace initial, ou espace d'entrée, vers la carte ou espace de sortie. À chaque neurone de la carte est associé un vecteur référent, appelé aussi vecteur prototype ou prototype, appartenant à l'espace d'entrée. En désignant par K le nombre total des neurones de la carte, le vecteur référent du neurone k est reconnu par $\boldsymbol{w}_k$ avec $k \in \{1, \ldots, K\}$ et $\boldsymbol{w}_k \in I\!R^p$. L'objectif de l'apprentissage de la carte consiste à mettre à jour les vecteurs référents de façon à approximer au mieux la distribution des vecteurs d'entrée tout en reproduisant l'auto-organisation des neurones de la carte. L'apprentissage de la carte se fait en mode séquentiel appelé aussi incrémental, ou en mode différé *(batch)*.

Apprentissage séquentiel

Chaque itération t de l'apprentissage séquentiel comprend deux étapes. La première étape consiste à choisir au hasard une observation $\boldsymbol{x}(t)$ de l'ensemble Ω, et à la présenter au réseau dans le but de déterminer son neurone vainqueur. Le neurone vainqueur *(Best Matching Unit)*, d'une observation est le neurone dont le vecteur référent en est le plus proche au sens d'une distance donnée (ex : distance euclidienne). Si c est le neurone vainqueur du vecteur $\boldsymbol{x}(t)$, c est déterminé comme suit :

$$d\big(\boldsymbol{w}_c(t), \boldsymbol{x}(t)\big) = \min_{k \in \{1,\ldots,K\}} d\big(\boldsymbol{w}_k(t), \boldsymbol{x}(t)\big) \tag{1.17}$$

Dans la deuxième étape, le neurone vainqueur est activé. Son vecteur référent est mis à jour pour se rapprocher du vecteur d'entrée présenté au réseau. Cette mise à jour ne concerne pas seulement le neurone vainqueur comme dans les méthodes de l'apprentissage par compétition *(Winner take all)*, mais aussi les neurones qui

lui sont voisins et qui voient alors leurs vecteurs référents s'ajuster vers ce vecteur d'entrée (voir figure 1.7, p.28). L'amplitude de cet ajustement est déterminée par la valeur d'un pas d'apprentissage $\alpha(t)$ et la valeur d'une fonction de voisinage $h(t)$. Le paramètre $\alpha(t)$ règle la vitesse de l'apprentissage. Il est initialisé avec une grande valeur au début puis décroît avec les itérations en vue de ralentir au fur et à mesure le processus d'apprentissage. La fonction $h(t)$ définit l'appartenance au voisinage. Elle dépend à la fois de l'emplacement des neurones sur la carte et d'un certain rayon de voisinage. Dans les premières itérations, le rayon de voisinage est assez large pour mettre à jour un grand nombre de neurones voisins du neurone vainqueur, mais ce rayon se rétrécit progressivement pour ne contenir que le neurone vainqueur avec ses voisins immédiats, ou bien même le neurone vainqueur seulement. La règle de mise à jour des vecteurs référents est la suivante (Kohonen, 1984; Ritter et Schulten, 1986) :

$$\boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) + \alpha(t) h_{ck}(t) \left[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)\right] \qquad k \in \{1, \ldots, K\} \tag{1.18}$$

où c est le neurone vainqueur du vecteur d'entrée $\boldsymbol{x}(t)$ présenté au réseau à l'itération t et $h_{ck}(t)$ est la fonction de voisinage qui définit la proximité entre les neurones c et k.

Une fonction de voisinage entre le neurone vainqueur c et un neurone k de la carte vaut 1 si le neurone k se trouve à l'intérieur du carré centré sur le neurone c et 0 dans les autres cas. Le rayon de ce carré est appelé rayon de voisinage. Il est large au début, puis se rétrécit avec les itérations pour contenir seulement le neurone c avec ses voisins immédiats à la fin de l'apprentissage ou même seulement le neurone c. Sur la figure 1.6, si le rayon de voisinage est 1, la fonction de voisinage vaut 1 pour les neurones contenus à l'intérieur du carré interne (vert) et 0 pour les autres neurones de la carte.

Une fonction de voisinage plus flexible et plus commune est la fonction gaussienne illustrée dans la figure 1.8 et définie ci-dessous :

$$\begin{aligned} h_{ck}\left(\sigma(t)\right) &= \exp\left(-\frac{d_2^2(\boldsymbol{r}_c, \boldsymbol{r}_k)}{2\sigma^2(t)}\right) \\ &= \exp\left(-\frac{\left\|\boldsymbol{r}_c - \boldsymbol{r}_k\right\|^2}{2\sigma^2(t)}\right) \end{aligned} \tag{1.19}$$

où $\boldsymbol{r}_c$ et $\boldsymbol{r}_k$ sont respectivement l'emplacement du neurone c et du neurone k sur la carte, et $\sigma(t)$ est le rayon du voisinage à l'itération t du processus d'apprentissage.

Avec une telle fonction de voisinage, l'amplitude de l'ajustement est graduée selon l'éloignement du neurone vainqueur qui réserve à lui-même l'amplitude maximale. Le résultat de cet apprentissage non supervisé est la projection non linéaire de l'ensemble des observations sur la carte. Chaque observation est attribuée à son neurone vainqueur. Outre la tâche de quantification, cette projection préserve la topologie des données grâce à l'utilisation de la fonction de voisinage. Deux neurones voisins sur la carte représenteront des observations proches dans l'espace de données.

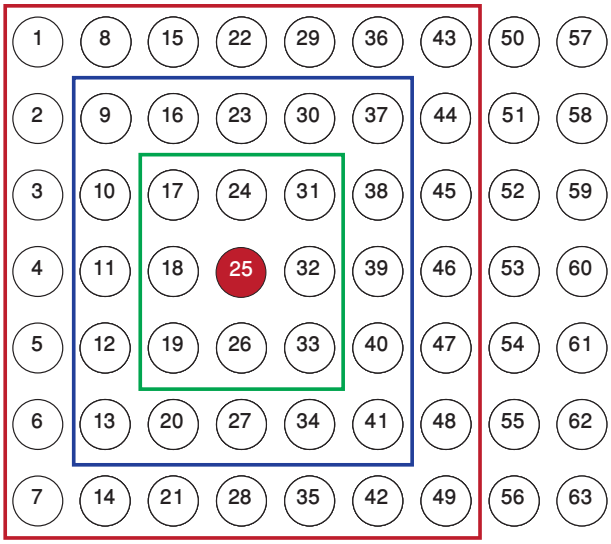


FIGURE 1.6 – Carte auto-organisatrice rectangulaire de 63 neurones.

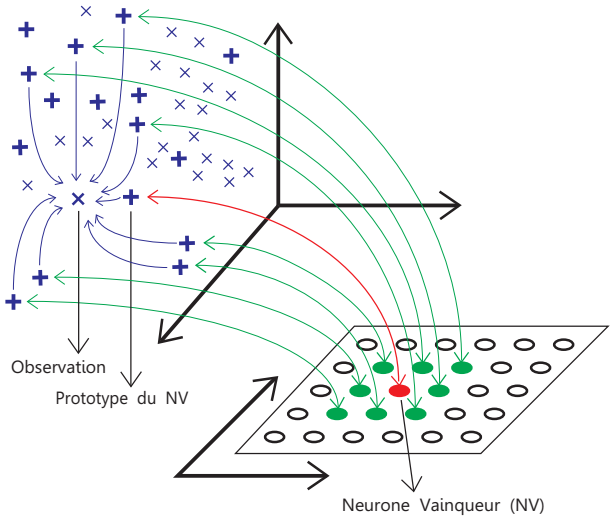


FIGURE 1.7 – Mise à jour des vecteurs prototypes.

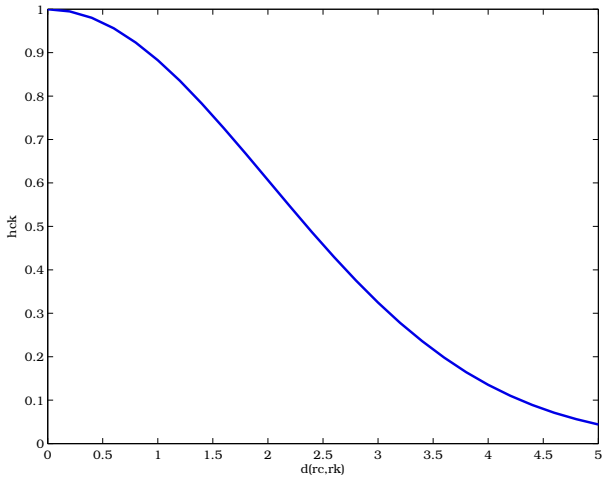


FIGURE 1.8 – Fonction de voisinage gaussienne.

L'algorithme 1.5 résume les étapes de l'apprentissage séquentiel des cartes auto-organisatrices.
La complexité d'un tel algorithme est $O(T.n.K.p)$ où T est le nombre total d'itérations, n le nombre d'observations, K le nombre de neurones et p la dimension des données. Cette complexité linéaire en fonction de n, de K ou de p rend l'utilisation d'un tel algorithme possible pour les ensembles de données contenant un grand nombre d'observations et à haute dimensionnalité.

Algorithme 1.5 Apprentissage séquentiel des cartes auto-organisatrices.

Entrées : Fournir :
- $\boldsymbol{\mathcal{X}}$ {□ La matrice des individus}
- lig, col {□ Dimensions de la carte (nombre de lignes, nombre de colonnes. $K = lig \cdot col$}
- T {□ Nombre total d'itérations}
- σ_{init}, σ_{final} {□ Valeurs initiale et finale du rayon de voisinage}
- α_{init} {□ Valeur initiale du pas d'apprentissage}

Sorties : Récupérer :
- $\boldsymbol{w}_k(T)$, $k \in \{1, \ldots, K\}$ {□ Les vecteurs prototypes auto-organisés}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Initialiser les vecteurs prototypes $\boldsymbol{w}_k(0)$ avec des valeurs aléatoires

Apprentissage :

Répéter
$\alpha(t) \leftarrow \alpha_{init}\left(1 - \frac{t}{T}\right)$
$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$
Choisir aléatoirement une observation $\boldsymbol{x}(t)$
Recherche de son neurone vainqueur (BMU) :
Pour $k = 1 \rightarrow K$ **Faire**
calculer $d_k \leftarrow d_2(\boldsymbol{x}, \boldsymbol{w}_k(t))$
Fin pour
$c \leftarrow \arg\min_k d_k$
Calcul des valeurs de la fonction de voisinage :
Pour $k = 1 \rightarrow K$ **Faire**
$h_{ck}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_c - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$
Fin pour
Mise à jour des vecteurs prototypes :
Pour $k = 1 \rightarrow K$ **Faire**
$\boldsymbol{w}_k(t+1) \leftarrow \boldsymbol{w}_k(t) + \alpha(t) h_{ck}(t) \left[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)\right]$
Fin pour
$t \leftarrow t + 1$
Jusqu'à $t > T$
Retourner $\boldsymbol{w}_k(T)$, $k \in \{1, \ldots, K\}$

Apprentissage en mode différé

En mode différé, à chaque itération t, toutes les observations sont présentées au réseau et la mise à jour des vecteurs prototypes se fait en prenant en compte toutes les observations de l'ensemble de données. Chaque vecteur prototype est une moyenne pondérée des vecteurs d'observations $(\boldsymbol{x}_i, i \in \{1, \ldots, n\})$ quand le carré de la distance euclidienne est utilisée pour le calcul du neurone vainqueur, les poids correspondants étant les valeurs de la fonction de voisinage $h(t)$ (Kohonen, 2001). La règle de mise à jour des vecteurs prototypes est donnée par :

$$\boldsymbol{w}_k(t+1) = \frac{\sum_{i=1}^{n} h_{kc_i}(t)\boldsymbol{x}_i}{\sum_{i=1}^{n} h_{kc_i}(t)} \qquad k \in \{1, \ldots, K\} \tag{1.20}$$

où h_{kc_i} est la valeur de la fonction de voisinage entre le neurone vainqueur c_i du vecteur $\boldsymbol{x}_i$ et le neurone k.

La mise à jour des vecteurs prototypes peut être formulée autrement en utilisant le fait que les observations qui ont le même neurone vainqueur ont la même valeur pour la fonction de voisinage et appartiennent à la région de Voronoï dont le centre est leur neurone vainqueur :

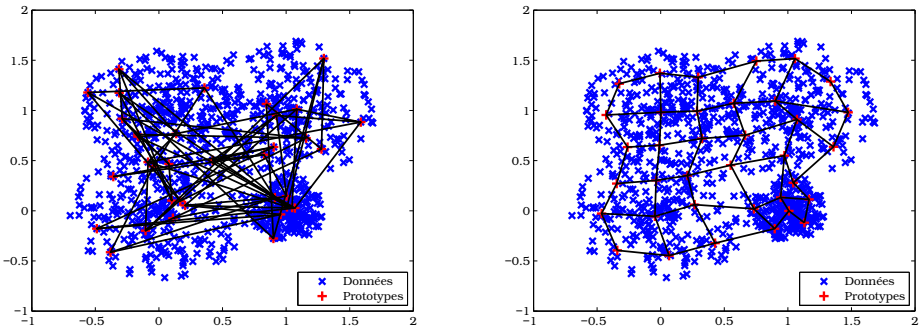
$$\boldsymbol{w}_k(t+1) = \frac{\sum_{l=1}^{K} h_{kl}(t) n_l \overline{x}_l}{\sum_{l=1}^{K} n_l h_{kl}(t)} \qquad k \in \{1, \ldots, K\} \tag{1.21}$$

où n_l est le nombre d'observations appartenant à la région de Voronoï représentée par le neurone l. et $\overline{x}_l$ est la moyenne des observations de cette même région.

Vers la fin de l'apprentissage, quand le rayon de voisinage devient trop petit pour activer seulement le neurone vainqueur, chaque vecteur prototype constitue le centre de gravité des observations qu'il représente et on retombe alors sur l'algorithme des centres-mobiles, ce qui garantit une meilleure approximation de la fonction de densité des observations (Kohonen, 2001). De plus, avec l'absence du pas d'apprentissage, cet algorithme ne présente pas de problèmes de convergence. Cependant, le mode différé pourrait causer des torsions dans les cartes à grandes dimensions. Pour cette raison, on procède à une analyse en composantes principales pour initialiser les vecteurs prototypes (voir sous-section 1.3.3, p.34).

La figure 1.9 trace les données et les prototypes avant et après apprentissage d'une carte auto-organisatrice carrée de 36 neurones pour un jeu de données à deux dimensions fourni par le logiciel Matlab sous le nom « *simplecluster_dataset.mat* ». Nous remarquons une bonne distribution des vecteurs prototypes sur les données ainsi qu'une carte bien ordonnée (chaque vecteur prototype se connecte aux vecteurs prototypes qui lui sont voisins), ce qui prouve que la topologie est préservée. Les prototypes initiaux sont choisis aléatoirement dans l'ensembles des observations et le nombre total d'itérations est de 200.

L'algorithme 1.6 présente les étapes de l'apprentissage en mode différé d'une carte auto-organisatrice. La complexité d'un tel algorithme est $O(T.K.n.p)$ où T est le nombre total d'itérations, K le nombre des neurones, n le nombre d'observations et p la dimension des données.



(a) Avant apprentissage (b) Après apprentissage

FIGURE 1.9 – Carte auto-organisatrice pour le jeu « *simplecluster_dataset.mat* ».

Algorithme 1.6 Apprentissage en mode différé des cartes auto-organisatrices.

Entrées : Fournir :

- $\boldsymbol{\mathcal{X}}$ {La matrice des individus}
- lig, col {Dimensions de la carte (nombre de lignes, nombre de colonnes. $K = lig \cdot col$}
- T {Nombre total d'itérations}
- σ_{init}, σ_{final} {Valeurs initiale et finale du rayon de voisinage}

Sorties : Récupérer :

- $\boldsymbol{w}_k(T)$, $k \in \{1, \ldots, K\}$ {□ Les vecteurs prototypes auto-organisés}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Initialiser les vecteurs prototypes $\boldsymbol{w}_k(0)$ avec des valeurs aléatoires

Apprentissage :

Répéter

$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$

Calculer les neurones vainqueurs des observations :

Pour $i = 1 \rightarrow n$ **Faire**

$d_2^2(\boldsymbol{x}_i, \boldsymbol{w}_{c_i}) \leftarrow \min_{k \in \{1,\ldots,K\}} d_2^2(\boldsymbol{x}_i, \boldsymbol{w}_k)$

Fin pour

Calcul des valeurs de la fonction de voisinage :

Pour $i = 1 \rightarrow n$ **Faire**

Pour $k = 1 \rightarrow K$ **Faire**

$h_{kc_i}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_i} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$

Fin pour
Fin pour
Mise à jour des vecteurs prototypes :
Pour $k = 1 \rightarrow K$ **Faire**
$\boldsymbol{w}_k(t) \leftarrow \frac{\sum_{i=1}^{n} h_{kc_i}(t)\boldsymbol{x}_i}{\sum_{i=1}^{n} h_{kc_i}(t)}$
Fin pour
$t \leftarrow t + 1$
Jusqu'à $t > T$
Retourner $\boldsymbol{w}_k(T)$, $k \in \{1, \ldots, K\}$

Évaluation de performance

Une carte auto-organisatrice est évaluée pour ses capacités de quantification et ses capacités de préservation de la topologie.

Pour mesurer le degré de déploiement de la carte sur les données ou le degré de quantification, on calcule la moyenne des erreurs de quantification *(Mean Quantization Error)* qui est définie par :

$$mqe = \frac{\sum_{i=1}^{n} d_2^2(\boldsymbol{x}_i, \boldsymbol{w}_{c_i})}{n} \tag{1.22}$$

où d_2^2 est le carré de la distance euclidienne et c_i le neurone vainqueur de l'observation $\boldsymbol{x}_i$.

Plusieurs critères existent pour quantifier la préservation de la topologie. Dans le cas où la carte est unidimensionnelle, on utilise souvent le critère suivant :

$$J = \sum_{k=2}^{K} d_2(\boldsymbol{w}_k, \boldsymbol{w}_{k-1}) - d_2(\boldsymbol{w}_K, \boldsymbol{w}_1) \tag{1.23}$$

La topologie est parfaitement préservée quand ce critère vaut 0.

Dans le cas général, un critère de préservation de la topologie sert à comparer les positions relatives des vecteurs référents par rapport aux positions relatives des neurones correspondants (Bauer et Pawelzik, 1992). Une autre approche consiste à mesurer le nombre de fois où la région de Voronoï d'un autre neurone de la carte s'introduit entre les vecteurs référents de deux unités voisines (Zrehen et Blayo, 1992).

Le critère que nous utilisons dans cette thèse mesure le nombre d'observations dont le premier neurone vainqueur (c_i) et le deuxième neurone vainqueur (ς_i) ne sont pas voisins sur la carte. Le deuxième neurone vainqueur d'une observation a son vecteur référent le plus proche de cette observation après celui du premier neurone vainqueur (Kiviluoto, 1996). Cette mesure s'appelle l'erreur topographique *(topographic error)* et est calculée comme suit :

$$te = \frac{\sum_{i=1}^{n} E}{n} \tag{1.24}$$

$$E = \begin{cases} 1 & \text{si } \|\boldsymbol{r}_{c_i} - \boldsymbol{r}_{\varsigma_i}\|^2 \neq 1 \\ 0 & \text{si } \|\boldsymbol{r}_{c_i} - \boldsymbol{r}_{\varsigma_i}\|^2 = 1 \end{cases}$$

Choix des paramètres du réseau

Il est souvent reproché aux cartes auto-organisatrices le nombre de paramètres à régler avant l'apprentissage, surtout qu'un mauvais choix de l'un de ces paramètres peut conduire à des résultats incohérents.

Les **vecteurs prototypes initiaux** ont une grande influence sur les résultats finaux au cas où ils sont initialisés aléatoirement. Une solution à ce problème consiste à exécuter plusieurs lancées de l'algorithme d'apprentissage, avec à chaque fois des vecteurs initiaux différents, et à adopter les résultats obtenus par la lancée qui permet de minimiser le plus l'erreur de quantification définie dans l'équation (1.22). Sinon, le choix des prototype initiaux peut se faire de manière déterministe en réalisant une analyse en composantes principales (ACP) des données. Les vecteurs prototypes seront alors positionnés sur les deux axes formés par les deux premiers vecteurs propres de la matrice de covariance des observations. Dans ce cas, l'initialisation des prototypes est linéaire. Une telle initialisation est à préconiser du fait qu'elle diminue le risque des torsions de la carte.

Le choix des **valeurs initiales et finales du rayon de voisinage et du pas d'apprentissage** pourraient avoir une influence sur les résultats obtenus. Pour les cartes à grandes dimensions, l'apprentissage se fait en deux phases. Dans la première phase, nommée phase d'organisation, le rayon de voisinage et le pas d'apprentissage sont initialisés avec de grandes valeurs et décroissent rapidement dans le but d'organiser les vecteurs prototypes. Par exemple, le rayon de voisinage prend une valeur qui vaut la moitié du rayon de la carte et le pas d'apprentissage est initialisé à 0.9.

La deuxième phase nommée phase de convergence a pour but d'ajuster les vecteurs prototypes vers leurs positions finales sans changer l'ordre qu'ils ont appris dans la phase précédente. Pour cette raison, le rayon de voisinage et le pas d'apprentissage sont initialisés avec de petites valeurs, par exemple ($\sigma_{init} = 2, \alpha_{init} = 0.1$). En général, le rayon de voisinage décroît pour atteindre la valeur 1, ce qui garantit une bonne préservation de la topologie. Cependant, ce rayon peut décroître au-dessous de 1 pour assurer un meilleur déploiement de la carte sur les données, surtout quand le but principal de l'utilisation de carte est la classification. Il est à noter que quand les prototypes sont initialisés moyennant une ACP, l'apprentissage se réduit à la phase de convergence seulement.

Critères de convergence

L'algorithme d'apprentissage des cartes auto-organisatrices dans sa version séquentielle ou en mode différé optimise le critère suivant (Ritter *et al.*, 1992; Ritter et Schulten, 1988) :

$$G_{SOM} = \frac{1}{2n} \sum_{i=1}^{n} \sum_{k=1}^{K} h_{kc_i} d_2^2(\boldsymbol{x}_i, \boldsymbol{w}_k) \tag{1.25}$$

Mais ce critère pose problème pour les observations qui se trouvent sur le bord du pavage de Voronoï et ayant plus qu'un neurone vainqueur, ce qui entraîne des points de discontinuités pour la fonction de voisinage h_{kc_i}. Cependant, au cas où les observations sont les réalisations d'une distribution discrète, la probabilité qu'une observation se trouve entre deux vecteurs prototypes sur le bord du pavage de Voronoi est nulle. Pour cette raison, il est possible d'adopter ce critère de convergence pour les ensembles de données finis et discrets.

Dans le cas où les données sont les réalisations d'une distribution continue, les cartes auto-organisatrices optimisent toujours le même critère mais à condition d'adopter la règle suivante pour la recherche du neurone vainqueur (Heskes, 1999a) :

$$c_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \sum_{l=1}^{K} h_{kl} d_2^2(\boldsymbol{x}_i, \boldsymbol{w}_l) \tag{1.26}$$

La convergence et la préservation de la topologie des cartes auto-organisatrices en mode différé ont été prouvées avec cette nouvelle règle pour rechercher le neurone vainqueur (Cheng, 1997).

L'apprentissage avec cette nouvelle règle pour la recherche du neurone vainqueur mène à des résultats très proches de ceux obtenus avec l'algorithme standard de Kohonen, mais nécessite des opérations de calcul plus complexes.

Cartes auto-organisatrices pour la classification

Les cartes auto-organisatrices sont souvent utilisées pour la classification non supervisée de données avec préservation de la topologie. À cette fin, chaque neurone représente une classe, et chaque observation est affectée au neurone dont le vecteur référent est le plus proche. De plus, deux neurones voisins sur la carte représenteront des observations qui sont proches dans l'espace d'entrée.

Pour avoir une meilleure distribution des vecteurs prototypes sur les données, le rayon de voisinage doit atteindre des valeurs assez petites pour n'activer que le neurone vainqueur. Le vecteur prototype correspondant sera alors le centre de gravité de la région de Voronoï qu'il représente. Donc, la classification moyennant les petites cartes se réduit à la méthode des centres-mobiles avec préservation de la topologie (Baçao *et al.*, 2005). La classification moyennant des cartes de petites dimensions présente plusieurs inconvénients notamment : la sensibilité de l'algorithme aux prototypes initiaux, le choix de la topologie de la carte qui est dictée par le nombre des classes et la difficulté de détecter des classes de formes quelconques.

Pour pallier ces inconvénients, des cartes de grandes dimensions sont utilisées, et la reconnaissance de classes de différentes formes peut se faire en inspectant la carte visuellement moyennant les *U-matrix* (Kohonen, 2001). Cependant, une telle détection des classes n'est pas toujours triviale, voire même impossible. Pour ces raisons, nous avons recours à des méthodes automatisées en classifiant les prototypes de la carte moyennant un algorithme de classification standard comme la classification hiérarchique ou la méthode des centres-mobiles (Vesanto et Alhoniemi, 2000). Il s'agit donc d'une classification en deux étapes : une projection des données sur une carte formée d'un grand nombre de neurones ($\approx 5\sqrt{n}$, par exemple) dans une première étape, suivie d'une classification des prototypes dans une deuxième étape où chaque observation appartiendra à la classe du vecteur prototype de son neurone vainqueur (voir figure 1.10, p.37). Cette technique présente plusieurs avantages :

1. Choix libre de la topologie de la carte :
   Le nombre de neurones n'est plus limité au nombre des classes.

2. Sensibilité réduite quant au choix des prototypes initiaux :
   Du fait que la carte auto-organisatrice constitue un résultat intermédiaire avant la classification finale, le problème de l'optimum local causé par le choix des prototypes initiaux est largement atténué (Dong et Xie, 2005).

3. Sensibilité réduite aux points aberrants :
   Les prototypes sont les moyennes des données et par suite ils sont moins sensibles aux variations extrêmes de ces données (Vesanto et Alhoniemi, 2000).

4. Réduction de la complexité algorithmique :
   Souvent, le nombre de classes est inconnu. Le programme de classification doit être exécuté plusieurs fois, avec à chaque fois un nombre de classe différent, afin de trouver le nombre de classes optimal au sens d'un critère interne (voir sous-section 1.4.2, p.41). Là, il vaut mieux projeter les données sur la carte et procéder ensuite à l'exécution du programme de classification plusieurs fois sur les prototypes de la carte pour pouvoir déterminer le nombre de classes qui convient le plus. Ce processus permet de réduire le temps de calcul car le nombre des prototypes est bien inférieur à celui des observations (Vesanto et Alhoniemi, 2000).

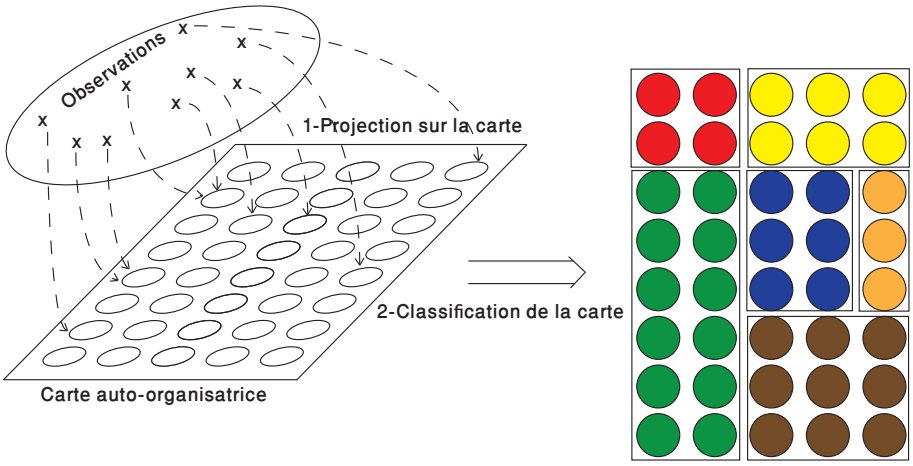


FIGURE 1.10 – Classification en deux étapes.

Variantes des cartes auto-organisatrices

Plusieurs variantes des cartes auto-organisatrices ont été proposées depuis leur conception. Nous en citons quelques unes dans ce qui suit :

Cartes auto-organisatrices arborescentes *(Tree Structured SOM).* La recherche séquentielle pour le neurone vainqueur peut nécessiter des calculs complexes et entraîner par la suite une lenteur dans l'apprentissage des cartes à dimensions importantes. Pour accélérer cette recherche, un système de recherche hiérarchique qui implique plusieurs cartes organisées en structure arborescente a été proposé. Chaque niveau de l'arbre se compose d'une carte plus développée que celle du niveau précédent. La recherche du neurone gagnant se fait niveau par niveau et à chaque fois la recherche concerne seulement un sous-ensemble de neurones qui sont descendants du neurone vainqueur du niveau précédent (Koikkalainen, 1995a).

Cartes auto-organisatrices hiérarchiques *(Hierarchical SOM).* Il s'agit de constituer une hiérarchie de plusieurs couches de cartes auto-organisatrices (Miikkulainen, 1990). La première couche contient seulement une carte. À chaque neurone de cette carte correspond une autre carte dans la couche suivante de la hiérarchie. Par exemple, en commençant avec une carte de 4 neurones dans la première couche, la deuxième couche sera composée de 4 cartes chacune ayant 4 neurones, pour finir avec la troisième couche qui contiendra 16 cartes de 4 neurones chacune. L'apprentissage des cartes se fait en commençant avec la carte de la première couche. Quand l'apprentissage touche à sa fin, les cartes de la couche suivante sont entraînées. Chaque carte sera entraînée seulement avec les observations qui ont comme neurone vainqueur, le neurone correspondant de la couche précédente. Ces types de cartes sont utilisées dans (Merkl, 1999) pour la classification des documents.

Neural Gas. Les *Neural Gas* sont des réseaux de neurones artificiels qui assurent la quantification vectorielle munie d'une structure de voisinage entre les neurones sans pour autant réaliser une projection de l'espace des observations à la manière des cartes auto-organisatrices classiques (Martinetz et Schulten, 1991). Par la suite, cette méthode requiert seulement le nombre de neurones sans avoir à fournir la topologie de la carte. La fonction de voisinage ne dépend pas de la distance entre deux neurones, mais depénd du rang du vecteur prototype dans la liste de vecteurs prototypes classés par ordre croissant de leur distance par rapport au vecteur d'entrée. La règle de mise jour des vecteurs prototypes suite à la présentation du vecteur $\boldsymbol{x}(t)$ au réseau est donnée par :

$$\boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) + \alpha(t) e^{\frac{-o_k}{\sigma(t)}} \big(\boldsymbol{x}_t - \boldsymbol{w}_k(t)\big) \tag{1.27}$$

où o_k est le rang du vecteur $\boldsymbol{w}_k$ tel que $o_k = 0$ si $\boldsymbol{w}_k$ est le plus proche de $\boldsymbol{x}(t)$ et $o_k = K - 1$ si $\boldsymbol{w}_k$ est le plus éloigné de $\boldsymbol{x}(t)$. $\alpha(t)$ (pas d'apprentissage) et $\sigma(t)$ (étendu de voisinage) décroissent avec les itérations. Après un nombre d'itérations suffisantes, l'ensemble des vecteurs prototypes constituera une représentation optimale de l'ensemble des observations.

Growing Neural Gas. L'algorithme *Growing Neural Gas* est similaire à l'algorithme *Neural Gas* à la différence que le nombre de neurones dans cette méthode n'est pas requis. L'algorithme commence seulement avec deux neurones et d'autres neurones seront ajoutés au fur et à mesure de façon à approximer au mieux la distribution des vecteurs d'entrée (Fritzke, 1995).

1.4 Évaluation d'une partition

Plusieurs critères existent pour évaluer le résultat d'une classification non supervisée générant une partition fixe des données. Ces critères sont de deux types : **externes** nécessitant des informations extérieures sur les classes, et **internes** s'appuyant seulement sur la structure interne des données.

1.4.1 Critères externes

Les critères externes sont basés sur des connaissances *a priori* des classes réelles et sont utilisés pour mesurer le degré de concordance entre la partition connue *a priori* et la partition issue du programme de classification. Nous citons ci-dessous deux critères externes utilisés couramment pour évaluer les résultats d'une classification.

Taux global d'erreur de classification *(Overall Error Rate of Classificaiton-OERC).* C'est le pourcentage des observations mal classifiées en supposant que l'appartenance des observations à leurs classes réelles est connue au préalable.
Soit $Q = \{L_1, \ldots, L_K\}$, la partition *a priori* et $P = \{C_1, \ldots, C_K\}$, la partition fournie par le programme de classification. Une classe de P peut correspondre à n'importe quelle classe de Q (par exemple C_2 correspond à L_3). Pour faire face à ce

problème, connu sous le nom de *switching problem*, la partition obtenue est comparée à toutes les $K!$ partitions générées par les permutations des K classes *a priori*. Seule la partition qui donne le plus grand nombre d'individus bien classés est choisie parmi les $K!$ partitions. L'utilisation de ce critère impose le même nombre de classes pour les deux partitions P et Q.

Indice de Rand *(Rand Index)* et Indice de Rand corrigé (Corrected Rand Index). Soit $Q = \{Z_1, \ldots, Z_L\}$, la partition *a priori* et $P = \{C_1, \ldots, C_K\}$ la partition fournie par le programme de classification. Le chevauchement entre les deux partitions P et Q est représenté par le tableau de contingence 1.2.

Tableau 1.2 – Tableau de contingence entre deux partitions.

P/Q	$\boldsymbol{Z_1}$	$\boldsymbol{Z_2}$	$\ldots$	$\boldsymbol{Z_L}$	**Total**
$\boldsymbol{C_1}$	n_{11}	n_{12}	$\ldots$	n_{1L}	$\boldsymbol{n_{1\cdot}}$
$\boldsymbol{C_2}$	n_{21}	n_{22}	$\ldots$	n_{2L}	$\boldsymbol{n_{2\cdot}}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\boldsymbol{C_K}$	n_{K1}	n_{K2}	$\ldots$	n_{KL}	$\boldsymbol{n_{K\cdot}}$
Total	$\boldsymbol{n_{\cdot 1}}$	$\boldsymbol{n_{\cdot 2}}$	$\ldots$	$\boldsymbol{n_{\cdot L}}$	$\sum_{kl} \boldsymbol{n_{kl}} = \boldsymbol{n}$

L'indice de Rand se base sur le nombre de paires d'observations pour lesquelles les deux partitions P et Q sont en accord ou en désaccord. Il est donné par :

$$RI = \frac{a+b}{a+b+c+d} \tag{1.28}$$

avec :

- a : le nombre de paires d'observations appartenant à la même classe dans P et dans Q.
- b : le nombre de paires d'observations qui appartiennent à des classes différentes dans P et dans Q.
- c : le nombre de paires d'observations qui appartiennent à la même classe dans P, mais appartiennent à des classes différentes dans Q.
- d : le nombre de paires d'observations qui appartiennent à des classes différentes dans P et à la même classe dans Q.

En effet, a et b sont utilisés comme indicateurs d'accord alors que c et d sont utilisés comme indicateurs de désaccord. Notons que $a+b+c+d$ est le nombre total

de paires d'observations, donc $a + b + c + d = \mathrm{C}_n^2$.
L'indice de Rand varie entre 0 et 1. Il vaut 1 en cas de parfait accord entre les 2 partitions n'ayant pas nécessairement le même nombre de classes.

Le problème avec l'indice de Rand est que sa valeur espérée n'est pas constante. La forme générale d'un indice avec une valeur espérée constante est : $\frac{indice - valeur\_esperee(indice)}{max(indice) - valeur\_esperee(indice)}$. L'indice de Rand Corrigé (Hubert et Arabie, 1985) a une valeur espérée qui vaut 0 et est donné par :

$$CR = \frac{\sum_{kl} \mathrm{C}_{n_{kl}}^2 - \left[\sum_k \mathrm{C}_{n_{k.}}^2 \sum_l \mathrm{C}_{n_{.l}}^2\right] / \mathrm{C}_n^2}{\frac{1}{2}\left[\sum_k \mathrm{C}_{n_{k.}}^2 + \sum_l \mathrm{C}_{n_{.l}}^2\right] - \left[\sum_k \mathrm{C}_{n_{k.}}^2 \sum_l \mathrm{C}_{n_{.l}}^2\right] / \mathrm{C}_n^2} \tag{1.29}$$

avec n_{kl}, $n_{.l}$ et $n_{k.}$ donnés par le tableau 1.2 et en supposant que :

$$E\left[\sum_{kl} \mathrm{C}_{n_{kl}}^2\right] = \frac{\left[\sum_k \mathrm{C}_{n_{k.}}^2 \sum_l \mathrm{C}_{n_{.l}}^2\right]}{\mathrm{C}_n^2} \tag{1.30}$$

L'indice de Rand corrigé varie entre -1 et +1, ce qui augmente sa sensibilité par rapport à l'indice de Rand. La valeur 1 indique un accord parfait entre les deux partitions, tandis qu'une valeur nulle ou négative indique une partition trouvée par chance.

D'autres critères externes existent pour comparer les résultats de deux classifications, entre autres : l'indice de Fowlkes-Mallows (Fowlkes et Mallows, 1983), l'indice *Entropy* et l'indice *Purity* (Zhao et Karypis, 2004).

Il à noter que l'algorithme de calcul des critères externes a une complexité algorithmique importante surtout quand le nombre de classes est assez élevé.

1.4.2 Critères internes

Dans beaucoup d'applications utilisant la classification non supervisée, aucune information sur les classes n'est connue *a priori*. Pour de tels cas, il existe des critères internes qui permettent d'évaluer les résultats d'une classification en utilisant des mesures de similarité entre les données. Nous citons par la suite deux critères internes bien connus en classification.

L'indice de Davies-Bouldin. L'indice de Davies-Bouldin (Davies et Bouldin, 1979) est fondé sur la mesure de similarité R_{kl} entre deux classes (C_k et C_l) calculée en tenant compte de la mesure de dispersion (s_k) de la classe C_k et de la mesure de dissimilarité ($D(C_k, C_l)$) entre les 2 classes. R_{kl} peut-être définie de n'importe quelle façon mais doit satisfaire les conditions suivantes :

- $R_{kl} \geqslant 0$
- $R_{kl} = R_{lk}$
- Si $s_k = 0$ et $s_l = 0$ alors $R_{kl} = 0$
- Si $s_l \geqslant s_q$ et $D(C_k, C_l) = D(C_k, C_q)$ alors $R_{kl} > R_{kq}$
- Si $s_l = s_q$ et $D(C_k, C_l) \leqslant D(C_k, C_q)$ alors $R_{kl} > R_{kq}$

Normalement, on définit R_{kl} de la façon suivante :

$$R_{kl} = \frac{s_k + s_l}{D(C_k, C_l)} \tag{1.31}$$

où $D(C_k, C_l) = d(\boldsymbol{w}_k, \boldsymbol{w}_l)$ est la distance entre les deux centres de gravités $\boldsymbol{w}_k$ et $\boldsymbol{w}_l$ des deux classes C_k et C_l, et s_k est la mesure de dispersion de la classe C_k calculée de la façon suivante :

$$s_k = \frac{\sum_{i=1}^{|C_k|} d(\boldsymbol{x}_i, \boldsymbol{w}_k)}{|C_k|} \tag{1.32}$$

où $|C_k|$ est le cardinal de classe C_k.

En général, la distance d doit correspondre à la distance utilisée dans le problème de classification.

L'indice Davies-Bouldin sera alors défini comme suit :

$$\begin{aligned} DB &= \frac{1}{K} \sum_{k=1}^{K} R_k \\ R_k &= \max_{\substack{l \in \{1,\ldots,K\} \\ l \neq k}} R_{kl} \end{aligned} \tag{1.33}$$

En comparant deux partitions, une valeur inférieure de cet indice signifie une meilleure classification. En effet, plus la valeur de cet indice est petite, plus les classes sont compactes et bien séparées.

Indice de Dunn. L'indice de Dunn (Dunn, 1973) vise à identifier des classes denses et bien séparées. Il est défini comme le rapport entre le minimum de distances

inter-classes et le maximum de distances intra-classes. Pour une partition donnée, l'indice de Dunn peut être calculé comme suit :

$$DI = \min_{k\in\{1,\ldots,K\}} \left\{ \min_{\substack{l\in\{1,\ldots,K\}\\ l\neq k}} \left\{ \frac{D(C_k, C_l)}{\max_{q\in\{1,\ldots,K\}} d'(C_k)} \right\} \right\} \tag{1.34}$$

où $D(C_k, C_l)$ est la distance inter-classe déterminée en calculant la distance entre les centres des 2 classes par exemple, et $d'(C_k)$ est la distance intra-classe mesurée en calculant la plus grande distance qui sépare deux observations de cette classe, ou en calculant la moyenne des distances séparant les observations du centre de leur classe. En comparant deux partitions, une valeur supérieure de cet indice signifie une meilleure classification. En effet, plus la valeur de cet indice est grande, plus les classes sont denses et bien séparées. Des indices de Dunn généralisés ont été proposés dans (Bezdek et Pal, 1998).

D'autres critères d'évaluation internes existent dans la littérature, à savoir : l'indice Silhouette (Rousseeuw, 1987), l'indice de Calinski-Harabasz (Calinski et Harabasz, 1974) et l'indice BIC (Bayesian Information Criterion) (Fraley et Raftery, 1998).

À part leur tâche d'évaluation de la qualité d'une partition, les critères internes sont souvent utilisés pour déterminer le nombre optimal de classes dans un problème de classification, en exécutant plusieurs fois le programme correspondant avec à chaque fois un nombre de classes différent, commençant par 2 classes et allant jusqu'à $\sqrt{n}$ classes. Par exemple, en utilisant l'indice Davies-Bouldin, le nombre de classes qui minimise le plus cet indice sera adopté (Vesanto et Alhoniemi, 2000).

1.5 Conclusion

Dans ce chapitre, nous avons évoqué les méthodes de classification non supervisées existantes en détaillant deux algorithmes utilisés dans nos travaux de recherche : l'algorithme des centres-mobiles et celui de la classification hiérarchique ascendante. Nous avons présenté aussi les critères utilisés pour évaluer numériquement le résultat d'une classification.

Nous nous sommes intéressés plus spécifiquement aux cartes auto-organisatrices de Kohonen du fait qu'elles constituent la base d'un bon nombre de travaux de recherche faits dans le cadre de cette thèse. Nous avons parlé de la théorie de ces

cartes et de leurs algorithmes d'apprentissage. Plus particulièrement, nous avons mis en évidence le rôle des cartes auto-organisatrices dans la classification non supervisée avec auto-organisation réalisée moyennant une petite carte où chaque neurone représente une classe, ou bien moyennant une grande carte où il faut procéder à la classification de la carte pour achever la classification finale des données.

Dans tout ce chapitre, les méthodes exposées s'appliquent à des données quantitatives où chaque observation est représentée par un point dans l'espace. Cependant, les données issues des expériences de la vie réelle sont souvent représentées sous plusieurs formes. Par exemple, une observation peut être de type qualitative, modale, intervalle, ou multi-valuée, d'où la nécessité de généraliser les méthodes classiques pour prendre en compte les différents types de données. Le chapitre 2 constitue une première migration des méthodes de classification non supervisée de leur formalisme classique vers leur formalisme symbolique.

Chapitre 2

Classification de données symboliques mixtes

2.1 Introduction

Dans les méthodes d'analyse de données classiques, les données sont modélisées par des tableaux à descriptions univaluées où chaque observation est représentée par un point dans l'espace $\mathbb{R}^p$. Or, un tel formalisme des données a un pouvoir expressif limité du fait qu'il ne tient pas compte des données complexes et hétérogènes issues des applications réelles, d'où la nécessité de modéliser les données avec des variables symboliques.

Une variable symbolique peut être représentée par une valeur unique quantitative ou qualitative, tout comme elle peut être aussi représentée par un intervalle de valeurs pour exprimer l'imprécision ou la variabilité des mesures, par un ensemble de valeurs dans le but d'exprimer le doute ou la multiplicité ou bien par d'autres formes. Donc, une variable symbolique peut avoir plusieurs représentations possibles visant à prendre en compte la complexité inhérente aux données.

L'analyse de données symboliques, ou *Symbolic Data Analysis (SDA)*, regroupe les techniques et les méthodes d'analyse de données qui permettent de traiter des données symboliques. Toutes ces méthodes résultent de l'extension des méthodes d'analyse classiques à des données exprimant un niveau de connaissance plus élevé qu'avec de simples observations. Les premiers travaux de recherche sur les méthodes d'analyse de données symboliques ont été proposées par Diday (1987, 1989). Depuis,

plusieurs méthodes d'analyse de données ont été étendues pour prendre en compte des données symboliques. Parmi ces méthodes, nous trouvons :

- La classification hiérarchique (Gowda et Diday, 1991; Ichino et Yaguchi, 1994; Chavent, 1997; Kim et Billard, 2012).
- L'analyse en composantes principales (Ichino et Yaguchi, 1994; Cazes *et al.*, 1997; Makosso K., 2010).
- La méthode des centres-mobiles (Ralambondrainy, 1995; Huang, 1997b).
- La méthode des nuées dynamiques (Chavent et Lechevallier, 2002; De Souza et De Carvalho, 2004; De Souza *et al.*, 2004; De Carvalho *et al.*, 2006; Verde et Irpino, 2007; De Carvalho et De Souza, 2010).
- Les réseaux de neurones (Rossi et Conan-Guez, 2002; Bock, 2003; El Golli *et al.*, 2004; Yang *et al.*, 2012).
- La classification floue (El-Sonbaty et Ismail, 1998; Yang *et al.*, 2004; De Carvalho, 2007).
- La régression (Billard et Diday, 2000; Domingues *et al.*, 2010).
- La classification basée sur les modèles de mélange : approche mélange (Hamdan et Govaert, 2003a,b, 2005) et approche classification (Hamdan et Govaert, 2004a,c).

Dans ce chapitre, nous allons parler des méthodes de classification de données symboliques. Nous débutons par une définition des données symboliques et des mesures existantes pour comparer deux vecteurs d'individus décrits par des variables symboliques. Puis, nous passons en revue les méthodes existantes de classification de données symboliques. Ensuite, nous proposons deux méthodes de classification de données symboliques en utilisant les cartes auto-organisatrices, pour finir avec une étude expérimentale qui permet de mettre en valeur les méthodes proposées et de les comparer à des méthodes existantes.

2.2 Données symboliques

Soit Ω un ensemble de n individus, ou observations, notés $\boldsymbol{X}_i$, $i \in \{1, \ldots, n\}$, décrits par p variables notées Y_j, $j \in \{1, \ldots, p\}$. Quand les valeurs des variables sont des nombres, elles sont appelées variables quantitatives. Mais, les données issues des applications réelles sont souvent plus complexes. Par exemple, nous pouvons trouver des valeurs textuelles, des ensembles de valeurs, des intervalles et bien d'autres

formes. Dans ce cas, les données sont symboliques et les observations sont décrites par des variables symboliques. Il est aussi très fréquent que les variables soient de différents types, il s'agit alors de variables symboliques mixtes.

On définit une variable Y_j comme une application de l'ensemble Ω dans le domaine U_j des valeurs possibles que peut prendre cette variable, qui à une observation $\boldsymbol{X}_i$ fait correspondre la valeur prise par la variable Y_j pour cette observation :

$$\begin{aligned} Y_j : \quad \Omega \quad &\longrightarrow U_j \\ \boldsymbol{X}_i \quad &\longrightarrow Y_j(\boldsymbol{X}_i) = X_i^j \end{aligned}$$

2.2.1 Types de variable symbolique

Dans cette thèse, nous traitons les variables symboliques ayant les types suivants :

1. **Quantitative univaluée :** c'est le cas des variables qui prennent des valeurs réelles uniques exprimant une quantité. $X_i^j \in U_j$, où $U_j \subset I\!R$.
   Exemple : $X_i^j = 150$ pour exprimer la taille d'une personne.

2. **Qualitative univaluée :** la variable prend des valeurs textuelles, aussi appelées modalités, dans le but d'exprimer une qualité. $X_i^j \in U_j$, où U_j est un ensemble de valeurs que peut prendre la variable Y_j.
   Exemple : $X_i^j = \{\text{bleu}\}$, où $U_j = \{\text{rouge, vert, bleu}\}$

3. **Intervalle :** ces types de variables sont utilisés pour exprimer la variabilité ou l'imprécision dans les données quantitatives. $X_i^j = [a_i^j, b_i^j]$ avec $a_i^j \in I\!R$, $b_i^j \in I\!R$ et $a_i^j \leqslant b_i^j$. $X_i^j \subset U_j = [a, b]$ où a et b sont respectivement la plus petite valeur et la plus grande valeur que peut prendre la variable Y_j.
   Exemple : $X_i^j = [165, 180]$ signifie que les tailles des étudiants d'une classe varient entre $165cm$ et $180cm$.

4. **Multivaluée nominale :** La variable prend plusieurs valeurs quantitatives ou qualitatives pour exprimer l'énumération ou le doute sans que ces valeurs soient ordonnées. $X_i^j \subset U_j$, où U_j est un ensemble fini de valeurs réelles ou textuelles que peut prendre la variable Y_j pour toutes les observations.

Exemple : $X_i^j = \{\text{bleu, rouge}\}$ et $U_j = \{\text{bleu, rouge, vert}\}$.

5. **Multivaluée ordinale :** si le domaine U_j est ordonné, la variable prend des valeurs liées par une relation d'ordre, comme un niveau de difficulté ({faible, moyen, fort}) ou un niveau d'études ({licence, master, doctorat}). Pour de telles descriptions, on peut coder numériquement les différentes valeurs, par exemple : $faible \leftarrow 1$, $moyen \leftarrow 2$, $fort \leftarrow 3$ dans le but de transformer ce genre de variables en variables quantitatives ou intervalle (Ichino et Yaguchi, 1994).
   Exemples :
   – Si $X_i^j = \{\text{faible}\}$ et $U_j = \{\text{faible, moyen, fort}\} \Rightarrow X_i^j = 1$ et $U_j = [1, 3]$

   – Si $X_i^j = \{\text{master, doctorat}\}$ et $U_j = \{\text{licence, master, doctorat}\} \Rightarrow X_i^j = [2, 3]$ et $U_j = [1, 3]$

6. **Modale ou de type histogramme** : c'est le cas d'une variable qualitative ou quantitative multivaluée associée de fréquences ou de poids pour les différentes valeurs de l'ensemble, formant ainsi un histogramme de valeurs. $X_i^j = \{F^j(i); \boldsymbol{\pi}^j(i)\}$ avec $F^j(i) \subset U_j$ et $\boldsymbol{\pi}^j(i)$ est le vecteur des fréquences dont les composantes ont des valeurs réelles. Quand les composantes sont normalisées, il s'agit alors de fréquences relatives appartenant à l'intervalle [0,1].
   Exemple :
   – $X_i^j = \big(\{\text{bleu, rouge, vert}\}; (2, 2, 1)^T\big)$

   – ou $X_i^j = \big(\{\text{bleu, rouge, vert}\}; (0.4, 0.4, 0.2)^T\big)$ en normalisant les fréquences.

   Les descriptions multivaluées sont un cas particulier de ce type correspondant à des fréquences valant 1 ou 0.

2.2.2 Définition d'un objet symbolique.

Un objet symbolique est défini pour représenter un ou plusieurs individus (Chavent, 1997). En analyse de données, plusieurs définitions d'objets symboliques sont suggérées. Un objet symbolique est défini par une conjonction logique des évène-

ments qui relient les variables et leurs valeurs. Nous donnons par la suite la définition d'un événement ainsi que la définition des objets assertions.

Définition d'un évènement. Un événement est défini par la paire *valeur-variable* et noté par $e_j = [Y_j = V_j]$ où V_j est la valeur prise par la variable Y_j ($V_j \subset U_j$). Exemple $e_1 = \big[Pays{=}\{\text{France, Japon}\}\big]$ est un évènement indiquant que la variable *pays* peut avoir soit la valeur « France » soit la valeur « Japon », et
$e_1 = \big[Taille = [165, 180]\big]$ est un évènement indiquant que la variable *Taille* a une valeur qui varie entre 165 et 180.

Définition d'un objet assertion. Un objet assertion est une conjonction logique d'événements définie par :

$$a = [Y'_1 = V_1] \wedge [Y'_2 = V_2] \wedge \ldots \wedge [Y'_q = V_q]$$

où le symbole $\wedge$ désigne une conjonction logique (non pas un produit vectoriel), $Y'_j \in \mathcal{Y} = \{Y_1, \ldots, Y_p\}$ et $V_j \subset U_j$. En fait, a est une application de l'ensemble Ω dans l'ensemble {Vrai, Faux}, qui fait correspondre la valeur « Vrai » à chaque individu ω vérifiant $Y'_j(\omega) \in V_j, \forall j$.
Exemple : soit l'objet assertion défini par :

$$a = [\text{Pays}{=}\{\text{France, Japon}\}] \wedge [\text{Taille} = [165, 180]] \wedge [\hat{\text{A}}\text{ge} = 25]$$

a est alors un objet assertion ayant les propriétés suivantes :

- La valeur de la variable *pays* est soit « France », soit « Japon ».
- La variable *taille* a une valeur est comprise entre 165 et 180
- La variable $\hat{A}ge$ vaut 25.

2.2.3 Tableau de données symboliques

Il s'agit d'un tableau dont les colonnes représentent les variables et les lignes les observations. Chaque cellule de ce tableau est la description d'une variable pour l'observation de la ligne correspondante. Le tableau 2.1 est un exemple d'un tableau de données symboliques. Il contient des informations concernant des classes d'étudiants. Chaque classe constitue une observation décrite par trois variables symboliques de différents types (Y_1 est de type qualitative multivaluée, Y_2 est de type intervalle et Y_3 est une variable de type histogramme).

Tableau 2.1 – Tableau de données symboliques.

Classe	**Groupe sanguin** (Y_1)	**Taille** (Y_2)	**Sexe** (Y_3)
Classe 1	{A^+, A^-, B^+}	$[150, 175]$	$(\{\text{fille, garçon}\}; (0.4, 0.6)^T)$
Classe 2	{A^-, B^-, B^+}	$[145, 170]$	$(\{\text{fille, garçon}\}; (0.3, 0.7)^T)$
Classe 3	{O^+, A^-, AB^+}	$[145, 170]$	$(\{\text{fille, garçon}\}; (0.3, 0.7)^T)$

Source de données symboliques. Les données symboliques sont généralement issues des expériences réelles, ou peuvent provenir de l'agrégation de grandes bases de données produisant ainsi de petits ensembles de données symboliques, ce qui constitue un avantage du fait que l'analyse de données sur les grands ensembles risque d'être lente, voire irréalisable. Le tableau 2.2 contient des informations concernant des étudiants représentés sous forme de 18 observations décrites avec des variables univaluées. En regroupant les informations par cours, nous obtenons le tableau 2.3 formé de 3 observations symboliques décrites par une variable modale et une variable de type intervalle.

Tableau 2.2 – Tableau de données à variables univaluées.

Étudiant	**Cours**	**Nationalité**	**Âge**
1	Programmation Web	Espagnol	22
2	Cobol	Français	25
3	Cobol	Japonais	24
4	Java	Japonais	25
5	Programmation Web	Espagnol	23
6	Cobol	Français	26
7	Java	Italien	27
8	Programmation Web	Italien	23
9	Java	Français	20
10	Cobol	Italien	24
11	Java	Japonais	20
12	Cobol	Français	30
13	Java	Japonais	22
14	Programmation Web	Italien	26
15	Cobol	Italien	25
16	Java	Japonais	20
17	Programmation Web	Italien	24
18	Java	Italien	22

Tableau 2.3 – Agrégation du tableau 2.2 par cours.

Cours	**Nationalité** (Y_1)	**Âge** (Y_2)
Programmation Web	$(\{\text{Espagnol, Italien}\};(2,3)^T)$	$[22,26]$
Cobol	$(\{\text{Français, Japonais, Italien}\};(3,1,2)^T)$	$[24,30]$
Java	$(\{\text{Italien, Français, Japonais}\};(2,1,4)^T)$	$[20,27]$

2.3 Mesures de proximité entre observations symboliques mixtes

La comparaison entre observations décrites par des variables symboliques doit souvent précéder toute tâche de classification à approche géométrique (qui utilisent des distances pour classifier les données). Dans le cas où les observations sont décrites par des variables symboliques de même type, le choix d'une mesure de rapprochement entre les individus est naturel. Par exemple, dans le cas où toutes les variables sont quantitatives univaluées, les distances de Minkowski pourront être utilisées pour mesurer le rapprochement entre observations symboliques (voir définition 1.4, p.13). De même, quand les variables sont toutes de type intervalle, on pourrait utiliser une distance basée sur la distance de Hausdorff pour comparer deux observations (voir équation (3.4), p.93). Toutefois, dans les applications réelles, il est fréquent de faire face à des données mixtes où les observations sont décrites par des variables de plusieurs types. Il faut avoir recours à des mesures plus élaborées pour réaliser cette comparaison. Par la suite, nous évoquerons les techniques et mesures de proximité existantes pour comparer des observations symboliques.

Dans le cas où les observations sont décrites par des variables symboliques de différents types, nous pourrons commencer par homogénéiser les données, dans le but de transformer un tableau de données hétérogènes en un tableau de données homogènes où les variables sont toutes de même type (De Souza *et al.*, 2007; De Carvalho et De Souza, 2010). Mais cette technique risque de provoquer une distorsion et une perte d'informations dans les résultats (Elghazel, 2007b). Une autre façon de faire est d'utiliser des indices de proximité qui permettent de comparer des objets où les variables sont de différents types, sans recodage préalable du tableau de données, comme la dissimilarité proposée par Gowda et Diday (1991) ou la distance proposée par Ichino et Yaguchi (1994).

2.3.1 Dissimilarité de Gowda et Diday

D'après Gowda et Diday (1991), une observation symbolique est le produit cartésien des valeurs prises par cette observation pour les différentes variables. $\boldsymbol{X}_i$ est alors exprimée comme suit :

$$\boldsymbol{X}_i = X_i^1 \times X_i^2 \times \ldots \times X_i^p \tag{2.1}$$

avec X_i^j la valeur que prend la variable Y_j pour l'observation $\boldsymbol{X}_i$, et U_j le domaine de la variable Y_j. La mesure de dissimilarité D entre les deux individus $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ est donnée par :

$$D\big(\boldsymbol{X}_i, \boldsymbol{X}_{i'}\big) = \sum_{j=1}^{p} D\big(X_i^j, X_{i'}^j\big)$$

avec

$$D\big(X_i^j, X_{i'}^j\big) = D_p\big(X_i^j, X_{i'}^j\big) + D_s\big(X_i^j, X_{i'}^j\big) + D_c\big(X_i^j, X_{i'}^j\big)$$

où :

- D_p est la composante de dissimilarité relative à la position. Elle est calculée seulement pour des variables quantitatives (intervalles, valeur unique), pour déterminer les positions relatives des valeurs que prend une variable pour deux individus sur la droite réelle.
- D_s est la composante de dissimilarité relative à la taille. Elle compare les étendues des intervalles ou les cardinaux des ensembles sans prendre en considération leur contenu.
- D_c est la composante de dissimilarité relative au contenu. Elle mesure la différence en contenu des deux valeurs que prend une variable.

Il est à noter que les trois composantes D_p, D_s et D_c sont définies telles qu'elles soient normalisées entre elles.

Variables quantitatives de type intervalle. Soient $X_i^j = [a_i^j, b_i^j]$ et $X_{i'}^j = [a_{i'}^j, b_{i'}^j]$ et soient :

- $inter_{ii'}^j$ la longueur de l'intervalle résultant de l'intersection des deux intervalles $[a_i^j, b_i^j]$ et $[a_{i'}^j, b_{i'}^j]$

– $ls_{ii'}^j$ la longueur de l'intervalle résultant de l'union jointe des deux intervalles $[a_i^j, b_i^j]$ et $[a_{i'}^j, b_{i'}^j]$ définie comme suit :

$$ls_{ii'}^j = \Big[\min\big(a_i^j, a_{i'}^j\big), \max\big(b_i^j, b_{i'}^j\big)\Big]$$

– l_i^j est la longueur de l'intervalle $[a_{i'}^j, b_{i'}^j]$ donnée par : $l_i^j = b_i^j - a_i^j$
la valeur de D_p est alors donnée par :

$$D_p\big(X_i^j, X_{i'}^j\big) = \frac{\big|a_i^j - a_{i'}^j\big|}{\big|U_j\big|}$$

où $\big|U_j\big|$ est la longueur de l'intervalle dont les bornes sont la valeur minimale et la valeur maximale du domaine U_j.

La valeur de D_s est donnée par :

$$D_s\big(X_i^j, X_{i'}^j\big) = \frac{\big|l_i^j - l_{i'}^j\big|}{ls_{ii'}^j}$$

La valeur de D_c est donnée par :

$$D_c\big(X_i^j, X_{i'}^j\big) = \frac{l_i^j + l_{i'}^j - 2 \cdot linter_{ii'}^j}{ls_{ii'}^j}$$

Variables modales ou de type histogramme. Soient $X_i^j = (F^j(i); \boldsymbol{\pi}^j(i))$ et $X_{i'}^j = (F^j(i'); \boldsymbol{\pi}^j(i'))$ deux variables modales telles que $F^j(i)$ est un ensemble de valeurs et $\boldsymbol{\pi}^j(i) = \big(\pi_1^j(i), \ldots, \pi_{Z_j}^j(i)\big)^T$ le vecteur des fréquences relatives correspondantes. Kim et Billard (2012) ont proposé une extension de la distance de Gowda et Diday pour pouvoir comparer deux variables modales. La composante de dissimilarité due à la position est absente. Donc, pour des variables identiques, nous nous contentons de calculer la composante relative due à la taille D_s et la composante relative due au contenu D_c.

$$D_s\big(X_i^j, X_{i'}^j\big) = \frac{\sum_{z=1}^{Z_j} \big|\pi_z^j(i) - \pi_z^j(i')\big|}{\sum_{z=1}^{Z_j} \pi_z^j(i \cup i')}$$

$$D_c = \frac{\sum_{z=1}^{Z_j} \big(\pi_z^j(i) + \pi_z^j(i') - 2 \cdot \pi_z^j(i \cap i')\big)}{\sum_{z=1}^{Z_j} \big(\pi_z^j(i) + \pi_z^j(i')\big)}$$

où $\pi_z^j(i \cup i') = \max\left(\pi_z^j(i), \pi_z^j(i')\right)$ et $\pi_z^j(i \cap i') = \min\left(\pi_z^j(i), \pi_z^j(i')\right)$

Ces composantes de dissimilarité ont toujours des valeurs entre 0 et 1. Elles valent 0 si les deux valeurs X_i^j, et $X_{i'}^j$ ont les mêmes fréquences et 1 si X_i^j et $X_{i'}^j$ ne se chevauchent pas.

Variables multivaluées qualitatives. Pour des variables identiques, la composante de dissimilarité due à la position étant absente, il suffit de calculer D_c et D_s. Soient :

- l_i^j le cardinal de l'ensemble X_i^j.
- $linter_{ii'}^j$ le cardinal de l'ensemble résultant de l'intersection des deux ensembles X_i^j et $X_{i'}^j$.
- $ls_{ii'}^j$ le cardinal de la réunion des ensembles X_i^j et $X_{i'}^j$ donné par : $ls_{ii'}^j = l_i^j + l_{i'}^j - linter_{ii'}^j$.

La composante de dissimilarité D_s est donnée par :

$$D_s\left(X_i^j, X_{i'}^j\right) = \frac{\left|l_i^j - l_{i'}^j\right|}{ls_{ii'}^j}$$

La composante de dissimilarité D_c est donnée par :

$$D_c\left(X_i^j, X_{i'}^j\right) = \frac{l_i^j + l_{i'}^j - 2 \cdot linter_{ii'}^j}{ls_{ii'}^j}$$

Variables univaluées quantitatives. Les variables quantitatives univaluées sont un cas particulier des variables de type intervalle où $a_i^j = b_i^j$ et $a_{i'}^j = b_{i'}^j$ et donc $l_i^j = l_{i'}^j = linter_{ii'}^j = 0$. La dissimilarité entre les deux valeurs quantitatives X_i^j et $X_{i'}^j$ est calculée par :

$$D\left(X_i^j, X_{i'}^j\right) = D_p(X_i^j, X_{i'}^j) = \frac{\left|a_i^j - a_{i'}^j\right|}{\left|U_j\right|}$$

Variables univaluées qualitatives. C'est un cas particulier des variables multivaluées nominales où la valeur d'une variable est un ensemble contenant seulement un élément. Donc, $l_i^j = l_{i'}^j = 1$ et $linter_{ii'}^j = 1$ si $X_i^j = X_{i'}^j$ et 0 sinon. La dissimilarité entre les deux valeurs qualitatives X_i^j et $X_{i'}^j$ est calculée comme suit :

$$D\left(X_i^j, X_{i'}^j\right) = D_c\left(X_i^j, X_{i'}^j\right) = \begin{cases} 1 & \text{si } X_i^j \neq X_{i'}^j \\ 0 & \text{si } X_i^j = X_{i'}^j \end{cases}$$

2.3.2 Distance de Minkowski généralisée pour les données symboliques mixtes

Ichino et Yaguchi (1994) ont généralisé la distance de Minkowski en la basant sur un nouveau modèle mathématique qu'ils ont défini et appelé le modèle de l'espace cartésien $(U_{(p)}, \boxplus, \boxtimes)$, où $U_{(p)}$ est l'espace p-dimensionnel des variables de différents types, $\boxplus$ un opérateur cartésien d'union jointe et $\boxtimes$ un opérateur cartésien d'intersection.

D'après Ichino et Yaguchi (1994), deux individus symboliques $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ sont définis par :

$$\boldsymbol{X}_i = X_i^1 \times X_i^2 \times \ldots \times X_i^p$$
$$\boldsymbol{X}_{i'} = X_{i'}^1 \times X_{i'}^2 \times \ldots \times X_{i'}^p$$

où X_i^j la valeur que prend la variable Y_j pour l'observation $\boldsymbol{X}_i$. U_j étant le domaine de la variable Y_j, alors $U_{(p)}$ est défini par :

$$U_{(p)} = U_1 \times U_2 \times \ldots \times U_p$$

Opérateur cartésien d'union jointe. L'union jointe entre deux observations $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ est définie par le produit suivant :

$$\boldsymbol{X}_i \boxplus \boldsymbol{X}_{i'} = (X_i^1 \boxplus X_{i'}^1) \times (X_i^2 \boxplus X_{i'}^2) \times \ldots \times (X_i^p \boxplus X_{i'}^p)$$

où $(X_i^j \boxplus X_{i'}^j)$ est le produit cartésien d'union jointe des valeurs que prend la variable Y_j pour les individus $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ défini comme suit :

– Si la variable Y_j est de type intervalle, les valeurs X_i^j et $X_{i'}^j$ sont alors deux intervalles tels que :

$$X_i^j = [a_i^j, b_i^j]$$
$$X_{i'}^j = [a_{i'}^j, b_{i'}^j]$$

le produit cartésien d'union jointe $(X_i^j \boxplus X_{i'}^j)$ est donc un intervalle fermé défini par :

$$X_i^j \boxplus X_{i'}^j = \left[\min\left(a_i^j, a_{i'}^j\right), \max\left(b_i^j, b_{i'}^j\right) \right]$$

Exemple : si $X_i^j = [2, 10]$ et $X_{i'}^j = [6, 14]$, alors $(X_i^j \boxplus X_{i'}^j) = [2, 14]$

– Si la variable Y_j est de type qualitative nominale, $(X_i^j \boxplus X_{i'}^j)$ est l'union de X_i^j et de $X_{i'}^j$:

$$(X_i^j \boxplus X_{i'}^j) = X_i^j \cup X_{i'}^j$$

Exemple : si $X_i^j = \{A,\ A^+,\ B^+\}$ et $X_{i'}^j = \{A,\ B^+,\ AB^+\}$, alors $(X_i^j \boxplus X_{i'}^j) = \{A,\ A^+,\ B^+,\ AB^+\}$

Opérateur cartésien d'intersection. L'intersection entre deux observations $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ est définie par le produit suivant :

$$\boldsymbol{X}_i \boxtimes \boldsymbol{X}_{i'} = (X_i^1 \boxtimes X_{i'}^1) \times (X_i^2 \boxtimes X_{i'}^2) \times \ldots \times (X_i^p \boxtimes X_{i'}^p)$$

où $(X_i^j \boxtimes X_{i'}^j)$ est le produit cartésien d'intersection des valeurs que prend la variable Y_j pour les individus $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ défini comme suit :

– Si la variable Y_j est de type intervalle, alors les valeurs X_i^j et $X_{i'}^j$ sont deux intervalles tels que :

$$\begin{aligned} X_i^j &= [a_i^j, b_i^j] \\ X_{i'}^j &= [a_{i'}^j, b_{i'}^j] \end{aligned} \tag{2.2}$$

alors le produit cartésien $(X_i^j \boxtimes X_{i'}^j)$ est un intervalle fermé défini par :

$$X_i^j \boxtimes X_{i'}^j = [a_i^j, b_i^j] \cap [a_{i'}^j, b_{i'}^j]$$

Exemple : si $X_i^j = [2, 10]$ et $X_{i'}^j = [6, 14]$, alors $(X_i^j \boxtimes X_{i'}^j) = [6, 10]$

– Si la variable Y_j est de type multivalueé, $(X_i^j \boxtimes X_{i'}^j)$ est l'intersection des deux ensembles X_i^j et $X_{i'}^j$:

$$(X_i^j \boxtimes X_{i'}^j) = X_i^j \cap X_{i'}^j$$

Exemple : si $X_i^j = \{A,\ A^+,\ B^+\}$ et $X_{i'}^j = \{A,\ B^+,\ AB^+\}$, alors $(X_i^j \boxtimes X_{i'}^j) = \{A,\ B^+\}$ où A, A^+, B^+, AB^+ sont des groupes sanguins.

Fonction de comparaison. La comparaison entre les deux individus $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ se réalise en commençant par comparer les valeurs que prennent les variables de l'un avec les valeurs que prennent les variables de l'autre deux à deux en utilisant une fonction de comparaison. Ensuite, la comparaison des deux observations est achevée en agrégeant les valeurs de ces fonctions de comparaison.

Une fonction de comparaison ϕ entre les valeurs que prend la variable Y_j pour l'individu $\boldsymbol{X}_i$ et pour l'individu $\boldsymbol{X}_{i'}$ est définie par :

$$\phi\big(X_i^j, X_{i'}^j\big) = \Big(\big|X_i^j \boxplus X_{i'}^j\big| - \big|X_i^j \boxtimes X_{i'}^j\big|\Big) + \gamma\Big(2 \cdot \big|X_i^j \boxtimes X_{i'}^j\big| - \big|X_i^j\big| - \big|X_{i'}^j\big|\Big)$$

où $\big|X_i^j\big|$ vaut la longueur de l'intervalle si Y_j est une variable de type intervalle, ou le nombre des éléments de l'ensemble X_i^j si Y_j est une variable multivaluée quantitative ou qualitative. Le paramètre γ est utilisé pour contrôler le rapprochement interne et le rapprochement externe entre deux intervalles. Par exemple, si dans un premier temps, $X_i^j = [2, 6]$ et $X_{i'}^j = [16, 20]$, en prenant $\gamma = 0$, la fonction de comparaison entre X_i^j et $X_{i'}^j$ devient :

$$\phi\big(X_i^j, X_{i'}^j\big) = \big|X_i^j \boxplus X_{i'}^j\big| - \big|X_i^j \boxtimes X_{i'}^j\big|$$

Or $\big|X_i^j \boxtimes X_{i'}^j\big| = 0$, ce qui donne : $\phi\big(X_i^j, X_{i'}^j\big) = 18$. Si dans un deuxième temps, $X_i^j = [2, 8]$ et $X_{i'}^j = [12, 20]$, en prenant $\gamma = 0$, on aura aussi $\phi\big(X_i^j, X_{i'}^j\big) = 18$, ce qui montre que si les intervalles sont disjoints, la fonction de comparaison ne prend en compte que le rapprochement externe en ignorant le rapprochement interne des intervalles (voir figure 2.1, p.58). Donc, le choix du paramètre γ doit se faire d'une façon judicieuse en fonction de l'analyse que l'on souhaite faire sur les données. Normalement, on choisit pour γ la valeur 0.5 (Ichino et Yaguchi, 1994) et la fonction de comparaison devient :

$$\phi\big(X_i^j, X_{i'}^j\big) = \big|X_i^j \boxplus X_{i'}^j\big| - \frac{\Big(\big|X_i^j\big| + \big|X_{i'}^j\big|\Big)}{2}$$

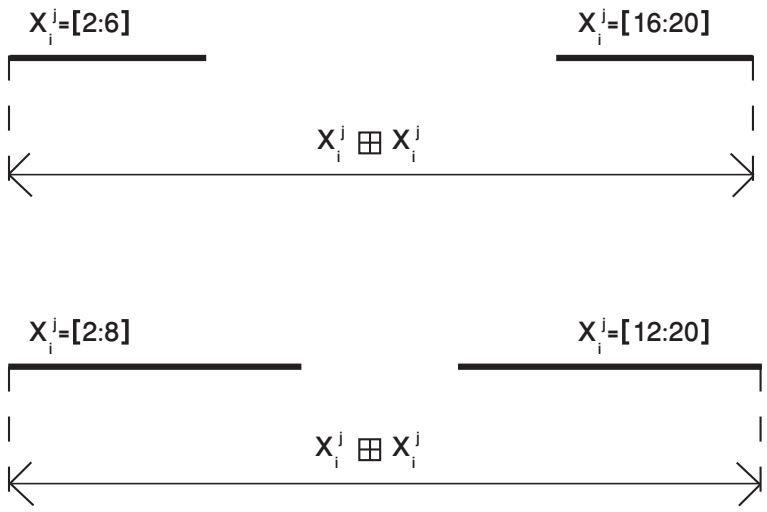


FIGURE 2.1 – Comparaison entre deux intervalles disjoints avec la distance de Ichino et Yaguchi.

La fonction ϕ vérifie les propriétés d'une distance (Ichino et Yaguchi, 1994) :

1. $\phi\big(X_i^j, X_{i'}^j\big) \geqslant 0 \, et \, \phi\big(X_i^j, X_{i'}^j\big) = 0 \Rightarrow X_i^j = X_{i'}^j$
2. $\phi\big(X_i^j, X_{i'}^j\big) = \phi\big(X_{i'}^j, X_i^j\big)$
3. $\phi\big(X_i^j, X_{i''}^j\big) \leqslant \phi\big(X_i^j, X_{i'}^j\big) + \phi\big(X_{i'}^j, X_{i''}^j\big)$

Comparaison entre deux observations symboliques. Finalement, la comparaison entre les deux observations $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ se fait en agrégeant les fonctions de comparaison pour toutes les variables et ceci en définissant la distance de Minkowski généralisée d'ordre q où $q \geqslant 1$:

$$dg_q\big(\boldsymbol{X}_i, \boldsymbol{X}_{i'}\big) = \Big[\sum_{j=1}^{p} \phi\big(X_i^j, X_{i'}^j\big)^q\Big]^{1/q}$$

Pour $q = 2$, on obtient la distance euclidienne généralisée, et pour $q = 1$, on obtient la distance city-block généralisée.

Ichino et Yaguchi (1994) ont proposé aussi de normaliser leur distance tout en affectant des poids aux différentes variables. Pour obtenir une distance normalisée, la fonction de comparaison est divisée par la longueur du domaine de la variable correspondante quand cette variable est un intervalle, ou par le nombre des éléments

du domaine de la variable quand cette dernière est multivaluée nominale. La distance normalisée est :

$$dg_q(\boldsymbol{X}_i, \boldsymbol{X}_{i'}) = \left[\sum_{j=1}^{p} \left(\zeta_j \psi\big(X_i^j, X_{i'}^j\big) \right)^q \right]^{1/q} \tag{2.3}$$

où ζ_j est le poids associé à la variable Y_j et $\psi\big(X_i^j, X_{i'}^j\big)$ est une fonction définie par :

$$\psi\big(X_i^j, X_{i'}^j\big) = \frac{\phi\big(X_i^j, X_{i'}^j\big)}{\big|U_j\big|}$$

Extension en cas de variables multimodales. Soient $X_i^j = (F^j(i); \boldsymbol{\pi}^j(i))$ et $X_{i'}^j = (F^j(i'); \boldsymbol{\pi}^j(i'))$ deux variables modales telles que $F^j(i)$ est un ensemble de valeurs et $\boldsymbol{\pi}^j(i) = \big(\pi_1^j(i), \ldots, \pi_{Z_j}^j(i)\big)^T$ le vecteur des fréquences relatives correspondantes. Kim et Billard (2012) ont proposé une extension de la distance d'Ichino et Yaguchi pour prendre en compte les variables modales.

$$\phi\big(X_i^j, X_{i'}^j\big) = \sum_{z=1}^{Z_j} \left\{ \big(\pi_z^j(i \cup i') - \pi_z^j(i \cap i')\big) + \gamma\big(2 \cdot \pi_z^j(i \cap i') - \pi_z^j(i) - \pi_z^j(i')\big) \right\}$$

où $\pi_z^j(i \cup i') = \max\big(\pi_z^j(i), \pi_z^j(i')\big)$ et $\pi_z^j(i \cap i') = \min\big(\pi_z^j(i), \pi_z^j(i')\big)$

Du fait que les poids varient entre 0 et 1, cette mesure est normalisée. Le paramètre γ varie entre 0 et 0.5. Il contrôle l'effet de la fréquence relative maximale et de la fréquence relative minimale entre les deux valeurs X_i^j et $X_{i'}^j$. Si la valeur de γ est nulle et si $\pi_z^j(i \cap i') = 0$ pour tout $z \in \{1, \ldots, Z_j\}$, il est possible alors de trouver plusieurs paires d'observations avec la même valeur de cette distance. D'autre part, la valeur 0.5 de γ annule l'effet de la portion commune entre deux observations ($\pi_z^j(i \cap i')$). Pour cela, une valeur de γ entre 0 et 0.5 est préconisée (Kim et Billard, 2012).

2.3.3 Autres mesures

D'autres mesures de proximité entre observations symboliques sont proposées dans la littérature dans le but d'améliorer les mesures existantes. Nous citons par la suite une modification de la dissimilarité de Gowda et Diday proposée par Yang *et al.* (2004), et les indices de proximité entre objets symboliques proposés par De Carvalho (1994).

Modification de la dissimilarité de Gowda et Diday. Yang *et al.* (2004) ont constaté que la dissimilarité de Gowda et Diday ne donne pas de bons résultats quand elle est utilisée comme mesure de proximité entre intervalles, et ceci en la comparant avec la distance de Hausdorff pour les intervalles. Ces résultats insatisfaisants sont causés par le fait que la composante de dissimilarité due à la position, met en jeu seulement les bornes inférieures des intervalles. Ils ont alors proposé remplacer la borne inférieure de chaque intervalle par son centre dans la composante D_p, et les trois composantes de dissimilarité pour comparer des intervalles deviennent :

$$\begin{aligned} D_p\big(X_i^j, X_{i'}^j\big) &= \frac{\Big|(a_i^j + b_i^j)/2 - (a_{i'}^j - b_{i'}^j)/2\Big|}{\big|U_j\big|} \\ D_s\big(X_i^j, X_{i'}^j\big) &= \frac{\big|l_i^j - l_{i'}^j\big|}{\Big(\big|U_j\big| + ls_{ii'}^j\Big)} \\ D_c\big(X_i^j, X_{i'}^j\big) &= \frac{l_i^j + l_{i'}^j - 2 \cdot linter_{ii'}^j}{\Big(\big|U_j\big| + ls_{ii'}^j\Big)} \end{aligned} \tag{2.4}$$

Rappelons que $\big|U_j\big|$ représente la longueur de l'intervalle formé par la borne inférieure et la borne supérieure du domaine U_j. Il est introduit dans le dénominateur de D_s et de D_c en vue de les normaliser.

Mesures de proximité entre objets symboliques. De Carvalho (1994) s'est basé sur la distance de Ichino et Yaguchi en proposant des indices de proximité entre deux objets symboliques a et b définis de la façon suivante :

$$\begin{aligned} a &= [Y_1 = A_1] \wedge [Y_2 = A_2] \wedge \ldots \wedge [Y_p = A_p] \\ b &= [Y_1 = B_1] \wedge [Y_2 = A_2] \wedge \ldots \wedge [Y_p = B_p] \end{aligned}$$

Quand a représente l'observation $\boldsymbol{X}_i$, les A_j sont remplacés par les X_i^j, pour $j \in \{1, \ldots, p\}$. La première mesure de proximité entre les objets a et b est donnée par :

$$d_{c1}(a, b) = \Pi(a \boxplus b) - \Pi(a \boxtimes b) + \gamma\big(2 \cdot \Pi(a \boxtimes b) - \Pi(a) - \Pi(b)\big)$$

où :

– $\Pi(a)$ est le potentiel de description de l'objet a défini par l'équation suivante :

$$\Pi(a) = \prod_{j=1}^{p} \mu(A_j)$$

$\mu(A_j)$ est défini par :

$$\mu(A_j) = \begin{cases} cardinal(A_j), \text{ si } Y_j \text{ est de type multivaluée} \\ Etendue(A_j), \text{ si } Y_j \text{ est de type intervalle} \end{cases}$$

- $a \boxplus b = [Y_1 = A_1 \cup B_1] \wedge [Y_2 = A_2 \cup B_2] \wedge \ldots \wedge [Y_p = A_p \cup B_p]$
- $a \boxtimes b = [Y_1 = A_1 \cap B_1] \wedge [Y_2 = A_2 \cap B_2] \wedge \ldots \wedge [Y_p = A_p \cap B_p]$

La deuxième mesure de proximité d_{c2} est une première version normalisée de la mesure d_{c1} définie par :

$$d_{c2} = \frac{\Pi(a \boxplus b) - \Pi(a \boxtimes b) + \gamma\big(2 \cdot \Pi(a \boxtimes b) - \Pi(a) - \Pi(b)\big)}{\Pi(a^{\Omega})}$$

où $a^{\Omega} = [Y_1 = U_1] \wedge [Y_2 = U_2] \wedge \ldots \wedge [Y_p = U_p]$

La troisième mesure de proximité est une deuxième version normalisée de la mesure d_{c1} définie par :

$$d_{c3} = \frac{\Pi(a \boxplus b) - \Pi(a \boxtimes b) + \gamma\big(2 \cdot \Pi(a \boxtimes b) - \Pi(a) - \Pi(b)\big)}{\Pi(a \boxplus b)}$$

Toutes ces mesures de proximité entre observations symboliques exposées ci-dessus, sont basées principalement sur la dissimilarité de Gowda et Diday ou sur la distance de Minkowski généralisée, connue aussi sous le nom de distance de Ichino et Yaguchi. Bien que ces deux mesures soient souvent utilisées dans la littérature, il convient de préciser que la présence du paramètre q dans la distance de Minkowski généralisée présente un avantage sur les autres mesures du fait qu'il permet de maîtriser la forme des classes que l'on désire obtenir. Par contre, un mauvais choix du paramètre γ peut conduire à des mesures inconvenables. Plus tard dans ce chapitre, nous allons présenter une méthode de classification à base de cartes auto-organisatrices pour les données symboliques mixtes, en utilisant la distance de Minkowski généralisée, que nous allons comparer à une autre méthode utilisant une version modifiée de la dissimilarité de Gowda et Diday. Dans ce qui suit, nous allons présenter les méthodes de partitionnement existantes pour les données symboliques.

2.4 Méthodes de partitionnement existantes pour les données symboliques

Plusieurs méthodes de classification ont été étendues dans le but de prendre en compte des données symboliques comme : la classification hiérarchique, la méthode

des centres-mobiles, la classification floue, la méthode des nuées dynamiques et les cartes auto-organisatrices. Dans ce qui suit, nous présentons les méthodes de classification par partitionnement existantes à base de centres-mobiles, à base de nuées dynamiques et à base de cartes auto-organisatrices.

2.4.1 Extension de l'algorithme des centres-mobiles

L'algorithme classique des centres-mobiles *(K-means)*, a été étendu pour prendre en compte des données quantitatives et qualitatives univaluées. Dans ce contexte, deux approches ont été proposées. La première approche effectue une transformation des données qualitatives en données binaires (Ralambondrainy, 1995). La deuxième approche propose une distance pour comparer des individus décrits par des variables quantitatives et qualitatives (Huang, 1997a).

Version conceptuelle des centres-mobiles. Ralambondrainy (1995) a appliqué l'algorithme des centres-mobiles à des données univaluées qualitatives et quantitatives. Dans cette approche, les valeurs des variables qualitatives sont codées numériquement et transformées en vecteurs de valeurs binaires, le nombre des composantes de chaque vecteur valant le nombre de modalités de la variable. Par exemple, pour la variable « Sexe », les modalités correspondantes sont : {mâle, femelle}, la valeur « mâle » est donc remplacée par le vecteur binaire $(1,0)^T$, et la valeur « femelle » par le vecteur $(0,1)^T$. Donc, quand les p variables sont qualitatives, une observation $\boldsymbol{X}_i$ est alors représentée par :

$$\boldsymbol{X}_i = (a_i^1, \ldots, a_i^m)^T \quad a_i^j \in \{0,1\}$$

m étant le nombre total des modalités. Afin de comparer deux observations $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ décrites par des variables qualitatives, on utilise la distance du Khi-deux définie par :

$$d^2_{\chi^2}(\boldsymbol{X}_i, \boldsymbol{X}_{i'}) = \sum_{j=1}^{m} \frac{(a_j - b_j)^2}{n_j}$$

n_j étant le nombre des observations contenant la $j^{\text{ème}}$ modalité. Cette distance donne ainsi plus d'importance aux modalités rares (parfois supposées apporter plus d'informations que celles plus fréquentes).

Dans le cas où les observations $\boldsymbol{X}_i$ et $\boldsymbol{X}_{i'}$ sont décrites exclusivement par des variables numériques, le carré de la distance euclidienne normalisée est utilisé pour

les comparer :

$$d^2_{2/\sigma^2}\big(\boldsymbol{X}_i, \boldsymbol{X}_{i'}\big) = \sum_{j=1}^{m} \frac{(a_j - b_j)^2}{\sigma_j^2}$$

où σ_j est l'écart-type de la $j^{\text{ème}}$ variable.

Quand les variables ne sont pas toutes de même type, Ralambondrainy a proposé la distance suivante pour comparer deux observations décrites par des variables quantitatives et qualitatives :

$$d^2\big(\boldsymbol{X}_i, \boldsymbol{X}_{i'}\big) = p_1 d^2_{\chi^2} + p_2 d^2_{2/\sigma^2}$$

où p_1 et p_2 sont des poids qui désignent l'importance que l'on désire attribuer aux variables qualitatives et quantitatives respectivement.

La conversion de données qualitatives en vecteurs binaires risque de produire des ensembles de données de grande dimension surtout quand le nombre de modalités est important. Un autre inconvénient de cette méthode réside dans le fait que les centres des classes auront des valeurs réelles comprises entre 0 et 1, ce qui n'assure pas une bonne représentation des classes.

K-prototypes et K-Modes. Huang (1997a) a proposé l'algorithme des K-prototypes pour classifier des données univaluées mixtes en se basant aussi sur l'algorithme des centres-mobiles mais sans transformation préalable des données qualitatives. Le but de l'algorithme proposé est de minimiser la fonction coût suivante :

$$E = \sum_{k=1}^{K} E_k = \sum_{k=1}^{K} \sum_{\boldsymbol{X}_i \in C_k} d\big(\boldsymbol{X}_i, \boldsymbol{Q}_k\big)$$

Dans le cas où le vecteur $\boldsymbol{X}_i$ est décrit par des variables mixtes quantitatives et qualitatives, la distance d entre une observation $\boldsymbol{X}_i$ et le prototype $\boldsymbol{Q}_k$ d'une classe k est défini par :

$$d\big(\boldsymbol{X}_i, \boldsymbol{Q}_k\big) = \sum_{j_r=1}^{p_r} d_2^2\big(X_i^{j_r}, w_k^{j_r}\big) + \gamma_k \sum_{j_c=1}^{p_c} \delta\big(X_i^{j_c}, q_k^{j_c}\big)$$

où :

- p_r est le nombre des variables quantitatives.
- p_c est le nombre des variables qualitatives.
- $X_i^{j_r}$ et $w_k^{j_r}$ sont les valeurs des variables quantitatives.

– $X_i^{j_c}$ et $q_k^{j_c}$ sont les valeurs des variables qualitatives.
– γ_k est le poids qu'on désire attribuer aux valeurs qualitatives de la classe C_k.
– δ est une distance entre deux valeurs qualitatives définie par :

$$\delta(M,N) = \begin{cases} 0 & \text{si } M = N \\ 1 & \text{si } M \neq N \end{cases}$$

La fonction coût E_k de la classe k devient alors :

$$E_k = \underbrace{\sum_{\boldsymbol{X}_i \in C_k} \sum_{j_r=1}^{p_r} d_2^2\big(X_i^{j_r}, w_k^{j_r}\big)}_{E_k^r} + \underbrace{\gamma_k \sum_{\boldsymbol{X}_i \in C_k} \sum_{j_c=1}^{p_c} \delta\big(X_i^{j_c}, q_k^{j_c}\big)}_{E_k^c}$$

E_k^r et E_k^c sont les fonctions coût de la classe C_k pour les variables numériques et qualitatives respectivement. Or, $w_k^{j_r}$ qui minimise E_k^r est le centre de gravité de la classe k, alors que $q_k^{j_c}$ qui minimise E_k^c est la modalité la plus choisie de la variable Y_j dans la classe C_k ou simplement le mode de Y_j dans l'ensemble C_k. Quand toutes les variables sont qualitatives, le calcul des prototypes des classes est réduit au calcul des modes et on retrouve l'algorithme des K-modes (Huang, 1997b) qui est une version simplifiée de l'algorithme des K-prototypes.

2.4.2 Méthode des nuées dynamiques

La méthode des nuées dynamiques constitue une généralisation de la méthode des centres-mobiles (voir sous-section 1.3.2, p.24). Cette méthode a été largement utilisée pour classifier des données symboliques. Par exemple, plusieurs chercheurs ont proposé des algorithmes de classification pour les données de type intervalle en se basant sur l'algorithme des nuées dynamiques. Ces méthodes seront évoquées dans le chapitre 3. Quand les observations sont décrites par des données symboliques mixtes, une technique d'homogénéisation des données est souvent requise dans le but de transformer les données de tous types en données de type histogramme (De Carvalho, 1995). L'algorithme des nuées dynamiques est alors exécuté sur cet ensemble de données modales où la distance euclidienne classique ou adaptative est utilisée pour mesurer la proximité entre les observations (De Souza *et al.*, 2007; De Carvalho et De Souza, 2010).

Il est possible de classifier des données symboliques mixtes avec l'algorithme des nuées dynamiques sans transformer les données au préalable et ceci en utilisant

des distances dépendantes du contexte comme la distance de Ichino et Yaguchi (voir sous-section 2.3.2, p.55) ou la dissimilarité de Gowda et Diday (voir sous-section 2.3.1, p.52). Dans de tels cas, le prototype d'une classe peut être représenté par l'observation de cette classe dont la somme des distances aux autres observations est minimale (Verde, 2004).

2.4.3 Cartes auto-organisatrices pour les dissimilarités

Les cartes auto-organisatrices permettent de classifier les données en préservant leur topologie (voir sous-section 1.3.3, p.25). Quand les données sont décrites par des variables symboliques de divers types, la détermination des vecteurs référents ou vecteurs prototypes d'une carte auto-organisatrice devient difficile. El Golli *et al.* (2004) ont proposé un algorithme d'apprentissage en mode différé (en mode *batch*) pour les cartes auto-organisatrices, appelé *DissSOM*, ayant comme entrée une matrice de dissimilarités ou de distances au lieu d'une matrice *individus-variables*, permettant ainsi de traiter des données de nature quelconque .

Étant donné n observations symboliques $\boldsymbol{X}_i$, $i \in \{1, \ldots, n\}$ appartenant à l'ensemble Ω, chaque neurone k est représenté par un individu référent $\boldsymbol{W}_k$, $k \in \{1, \ldots, K\}$ constitué par un ensemble contenant quelques observations de l'ensemble Ω tel que :

$$\boldsymbol{W}_k = \{\boldsymbol{X}_{v1}, \boldsymbol{X}_{v2}, \ldots, \boldsymbol{X}_{v\ell}\}$$

avec $\boldsymbol{X}_{vs} \in \Omega$ pour tout $s \in \{1, \ldots, \ell\}$. Les auteurs définissent une dissimilarité généralisée d^T de $\Omega \times P(\Omega)$ dans $\mathbb{R}^+$ qui permet de mesurer le rapprochement entre une observation $\boldsymbol{X}_i$ et un individu référent $\boldsymbol{W}_k$ par :

$$d^T\big(\boldsymbol{X}_i, \boldsymbol{W}_k\big) = \sum_{r=1}^{K} h_{kr}(t) \sum_{s=1}^{\ell} d^2\big(\boldsymbol{X}_i, \boldsymbol{X}_{vs}\big) \text{ tel que } \boldsymbol{X}_{vs} \in \boldsymbol{W}_r$$

où $h_{kr}(t)$ est la fonction de voisinage gaussienne entre les neurones k et r à l'itération t (voir équation (1.19), p.27).

L'objectif de l'apprentissage de la carte est de minimiser la fonction coût suivante :

$$E = \sum_{i=1}^{n} d^T\big(\boldsymbol{X}_i, \boldsymbol{W}_{c_i}\big) = \sum_{i=1}^{n} \sum_{k=1}^{K} h_{kc_i}(t) \sum_{s=1}^{\ell} d^2\big(\boldsymbol{X}_i, \boldsymbol{X}_{vs}\big) \text{ tel que } \boldsymbol{X}_{vs} \in \boldsymbol{W}_k$$

où c_i est le neurone vainqueur de l'observation $\boldsymbol{X}_i$, déterminé comme suit :

$$c_i = \underset{k\in\{1,\ldots,K\}}{\arg\min}\, d^T\big(\boldsymbol{X}_i, \boldsymbol{W}_k\big) \tag{2.5}$$

La détermination d'un individu référent $\boldsymbol{W}_k$ se fait en minimisant la fonction coût E_k :

$$E_k = \sum_{i=1}^{n} h_{kc_i} \sum_{s=1}^{\ell} d^2\big(\boldsymbol{X}_i, \boldsymbol{X}_{vs}\big) \text{ tel que } \boldsymbol{X}_{vs} \in \boldsymbol{W}_k \tag{2.6}$$

Chaque itération t de l'algorithme *DissSOM* consiste en une **phase d'affectation**, où le neurone vainqueur de chaque observation symbolique est déterminé suivant l'équation (2.5), et en une **phase de représentation** où les individus référents sont calculés en minimisant l'équation (2.6). Le rayon de voisinage (l'écart-type de h_{kc_i}) décroît à chaque itération pour réduire progressivement le voisinage d'un neurone vainqueur.

2.4.4 Cartes auto-organisatrices pour les chaînes de caractères

Il est possible d'utiliser les cartes auto-organisatrices pour des chaînes de caractères *(SOM for Symbol String)* (Kohonen, 2001). Pour des données identiques, l'apprentissage de la carte se fait en mode différé en utilisant une distance qui permet de comparer deux chaînes de caractères, comme la distance de Levensthein. Le prototype d'un neurone vainqueur vaut la médiane généralisée de l'ensemble des observations qu'il représente. Nous donnons par la suite une définition de la distance de Levensthein et de la médiane généralisée pour un ensemble, pour ensuite exposer les étapes de l'algorithme d'apprentissage.

Distance de Levensthein. La distance de Levensthein entre deux chaînes de caractères ($Str1$ et $Str2$) est le nombre minimal de caractères à remplacer, insérer et supprimer pour transformer $Str1$ en $Str2$. Exemple : $Str1$=« danse » et $Str2$=« lancée », la distance de Levenshtein entre $Str1$ et $Str2$ vaut 4 du fait que les opérations minimales requises pour transformer $Str1$ en $Str2$ sont :

1. Remplacer 'd' par 'l'.
2. Remplacer 's' par 'c'.

3. Remplacer 'e' par 'é'.
4. Insérer 'e'.

Médiane généralisée d'un ensemble de chaînes de caractères. Afin de déterminer la médiane généralisée d'un ensemble de chaînes de caractères, on commence tout d'abord par déterminer la médiane d'un ensemble de chaînes de caractères en calculant toutes les distances mutuelles entre les différentes chaînes et en choisissant celle ayant la plus petite somme de distances des autres chaînes de l'ensemble. Ensuite, sur cette médiane, plusieurs opérations sont effectuées (remplacement, insertion, suppression) mettant en jeu tous les caractères de l'alphabet, et seulement les opérations qui réduisent davantage la somme de distances des autres chaînes sont prises en compte, conduisant alors à l'obtention de la médiane généralisée (Kohonen, 2001).

Algorithme d'apprentissage de la carte auto-organisatrice. Les étapes de l'algorithme d'apprentissage des cartes auto-organisatrices pour les chaînes de caractères sont :

1. **Initialiser les prototypes des neurones**, soit avec des chaînes de caractères choisies au hasard, soit en projetant quelques chaînes à l'aide de la méthode de Sammon, et de choisir ensuite celles qui paraissent ordonnées d'une manière bi-dimensionnelle, comme prototypes initiaux.
2. **Déterminer le neurone vainqueur de chaque chaîne de caractères.** C'est le neurone dont le prototype est le plus proche de cette chaîne au sens d'une distance donnée, par exemple, la distance de Levenshtein.
3. **Mettre à jour les prototypes des neurones.** Le nouveau prototype du neurone k sera la médiane généralisée de l'ensemble des chaînes de caractères ayant le neurone k et les neurones appartenant à son voisinage N_k comme neurones vainqueurs.
4. **Répéter l'étape 2** jusqu'à ce que les prototypes ne changent plus.

Remarque concernant le voisinage N_k d'un neurone k. Au début de l'algorithme, ce voisinage est assez grand si les prototypes initiaux sont choisis aléatoirement. Par contre, il est limité si le choix des prototypes se fait en projetant

les données et en leur affectant des chaînes ordonnées. À chaque itération de l'algorithme, ce voisinage est réduit progressivement pour contenir enfin le neurone vainqueur, avec ou sans ses voisins immédiats, à la terminaison de l'algorithme.

Dans ce qui suit, nous allons présenter deux approches à base de cartes auto-organisatrices dans le but de classifier des données symboliques mixtes.

2.5 Cartes auto-organisatrices pour les données symboliques

Dans ce chapitre, nous proposons deux approches pour créer une carte auto-organisatrice pour des données symboliques mixtes. Dans la première approche, nous procédons à une homogénéisation des données en première étape où toutes les variables sont transformées en variables de type histogramme. L'ensemble de données ainsi obtenu est utilisé en deuxième étape pour entraîner une carte auto-organisatrice en mode différé en utilisant la distance euclidienne pour la recherche du neurone vainqueur. La deuxième approche consiste à entraîner une carte auto-organisatrice en se basant sur l'algorithme *DissSOM* décrit dans la sous-section 2.4.3. Cette méthode consiste à présenter une matrice de dissimilarités ou de distances à la carte en vue de l'entraîner en mode différé. La distance proposée par Ichino et Yaguchi (voir sous-section 2.3.2, p.55) est utilisée pour construire la matrice des distances.

2.5.1 1ère approche : Carte auto-organisatrice pour des données homogénéisées

Nous détaillons par la suite la technique d'homogénéisation des données, comme nous présentons l'algorithme d'apprentissage de la carte en utilisant les données transformées.

Homogénéisation des données symboliques mixtes

Nous adoptons la technique proposée par De Carvalho (1995) permettant de convertir des variables multivaluées et des variables de type intervalle en des variables de type histogramme ou modales dans le but de produire un ensemble de données

homogènes.

Transformation des variables multivaluées. Supposons que la variable Y_j est une variable multivaluée ayant pour domaine $U_j = \{f_1^j, \ldots, f_{Z_j}^j\}$ telle que $X_i^j \subset U_j$. Y_j est transformée en une variable de type histogramme H_j telle que $H_j(i) = \big(U_j; \boldsymbol{\pi}_j(i)\big)$ où $\boldsymbol{\pi}_j(i) = \big(\pi_1^j(i), \pi_2^j(i), \ldots, \pi_{Z_j}^j(i)\big)^T$ est le vecteur des fréquences relatives calculées de la façon suivante :

$$\pi_z^j(i) = \begin{cases} \frac{1}{\left|X_i^j\right|} & \text{si } f_z^j \in X_i^j \\ 0 & \text{if } f_z^j \notin X_i^j \end{cases} \tag{2.7}$$

où $z \in \{1, \ldots, Z_j\}$ et $\left|X_i^j\right|$ est le cardinal de l'ensemble X_i^j.

Transformation des variables de type intervalle. Soit Y_j une variable de type intervalle telle que $X_i^j = [a_i^j, b_i^j]$. Y_j est transformée en une variable de type histogramme cumulé H_j telle que : $H_j(i) = (U_j; \boldsymbol{\pi}_j(i))$ où $U_j = \{I_1^j, \ldots, I_{Z_j}^j\}$ est un ensemble d'intervalles élémentaires obtenu en formant une liste triée par ordre croissant de toutes les bornes inférieures et supérieures de tous les intervalles de cette variable pour toutes les observations. Chaque paire de nombres consécutifs de cette liste triée constituera les bornes d'un intervalle élémentaire. $\boldsymbol{\pi}_j(i) = \big(\pi_1^j(i), \pi_2^j(i), \ldots, \pi_{Z_j}^j(i)\big)^T$ est le vecteur des poids cumulatifs ayant comme support U_j et dont les composantes sont calculées comme suit :

$$\pi_z^j(i) = \sum_{v=1}^{z} u_v(i)^j \text{ tel que } u_v(i)^j = \frac{L(I_v^j \cap X_i^j)}{L(X_i^j)}$$

où $z \in \{1, \ldots, Z_j\}$ et $L(I)$ la longueur de l'intervalle I.

Exemple d'homogénéisation des données. En appliquant la technique d'homogénéisation décrite ci-dessus sur le tableau 2.3, nous obtenons le tableau 2.4 de la page 70 où les données sont décrites par des variables de type histogramme. Pour transformer la variable de type intervalle Y_2 en une variable de type histogramme H_2, il faut tout d'abord déterminer l'ensemble des intervalles élémentaires qui constituent le support des fréquences cumulatives. La liste triée des bornes inférieures et supérieures de tous les intervalles de cette variable vaut : {20, 22, 24, 26, 27, 30},

donc les intervalles élémentaires sont : $I_1^2 = [20, 22[$, $I_2^2 = [22, 24[$, $I_3^2 = [24, 26[$, $I_4^2 = [26, 27[$, $I_5^2 = [27, 30]$ formant ainsi l'ensemble $U_2 = \{I_1^2, I_2^2, I_3^2, I_4^2, I_5^2\}$. Pour la variable modale Y_1, le domaine correspondant est l'ensemble $U_1 = \{$Espagnol, Français, Japonais, Italien$\}$ et le vecteur des fréquences relatives pour la première observation est $\boldsymbol{\pi}_1(1) = (0.4, 0, 0, 0.6)^T$.

Tableau 2.4 – Tableau de données symboliques homogènes.

Cours	**Nationalité** (Y_1)	**Âge** (Y_2)
Programmation Web	$(U_1; \boldsymbol{\pi}_1(1) = (0.4, 0, 0, 0.6)^T)$	$(U_2; \boldsymbol{\pi}_2(1) = (0, 0.5, 1, 1, 1)^T)$
Cobol	$(U_1; \boldsymbol{\pi}_2(1) = (0, 0.5, 0.167, 0.333)^T)$	$(U_2; \boldsymbol{\pi}_2(2) = (0, 0, 0.3333, 0.5, 1)^T)$
Java	$(U_1; \boldsymbol{\pi}_3(1) = (0, 0.143, 0.571, 0.286)^T)$	$(U_2; \boldsymbol{\pi}_2(3) = (0.286, 0.571, 0.857, 1, 1)^T)$

Algorithme d'apprentissage de la carte auto-organisatrice

Partant d'un ensemble de données décrites par des variables symboliques mixtes, chaque observation $\boldsymbol{X}_i$ est transformée en un vecteur $\tilde{\boldsymbol{x}}_{\boldsymbol{i}}$ dont les composantes sont les fréquences attribuées aux différentes modalités lors de la transformation des variables symboliques $Y_j, j \in \{1, \ldots, p\}$ multivaluées ou intervalles en des variables de type histogramme $H_j, j \in \{1, \ldots, p\}$. Par la suite, nous utilisons les notations suivantes :

- $H_j(i) = \tilde{x}_i^j$
- $\boldsymbol{x}_i^j = \boldsymbol{\pi}_j(i)$
- $x_{iz}^j = \pi_z^j(i)$
- $\boldsymbol{x}_i^j = (x_{i1}^j, x_{i2}^j, \ldots, x_{iZ_j}^j)^T$
- $\tilde{x}_i^j = \left(U_j; \boldsymbol{x}_i^j\right)$

Le tableau 2.5 représente l'ensemble de données homogénéisées où $f_z^j, z \in \{1, \ldots, Z_j\}$ est un élément de l'ensemble constitué par le domaine de la variable multivaluée Y_j, ou un intervalle élémentaire si la variable Y_j est de type intervalle.

Tableau 2.5 – Tableau de données de type histogramme.

variables	H_1				...	H_p			
modalités	f_1^1	f_2^1	...	$f_{Z_1}^1$	...	f_1^p	f_2^p	...	$f_{Z_p}^p$
$\tilde{\boldsymbol{x}}_{\mathbf{1}}$	x_{11}^1	x_{12}^1	...	$x_{1Z_1}^1$	...	x_{11}^p	x_{12}^p	...	$x_{1Z_p}^p$
$\tilde{\boldsymbol{x}}_{\mathbf{2}}$	x_{21}^1	x_{22}^1	...	$x_{2Z_1}^1$	...	x_{21}^p	x_{22}^p	...	$x_{2Z_p}^p$
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
$\tilde{\boldsymbol{x}}_{\boldsymbol{n}}$	x_{n1}^1	x_{n2}^1	...	$x_{nZ_1}^1$	...	x_{n1}^p	x_{n2}^p	...	$x_{nZ_p}^p$

Supposons que nous voulions créer une carte auto-organisatrice composée de K neurones pour classifier l'ensemble de données $\tilde{\boldsymbol{x}}_{\boldsymbol{i}}, i \in \{1, \ldots, n\}$. Le prototype $\boldsymbol{g}_k$ de chaque neurone k est aussi représenté par un vecteur de données décrites par des variables histogrammes, le $j^{\text{ème}}$ histogramme du neurone k est noté $g_k^j = (U_j; \boldsymbol{w}_k^j)$ (U_j étant le domaine de la variable H_j), et $\boldsymbol{w}_k^j = (w_{k1}^j, \ldots, w_{kZ_j}^j)^T$.

Chaque itération de l'algorithme d'apprentissage en mode différé consiste à présenter tout l'ensemble de données au réseau, à déterminer le neurone vainqueur de chaque observation, pour ensuite mettre à jour les vecteur prototypes de l'ensemble des neurones.

Détermination du neurone vainqueur. Le neurone vainqueur d'un vecteur de données $\tilde{\boldsymbol{x}}_i$ est le neurone dont le vecteur prototype $\boldsymbol{g}_{c_i}$ est le plus proche de $\tilde{\boldsymbol{x}}_i$ au terme du carré de la distance euclidienne :

$$
\begin{aligned}
d_2^2(\tilde{\boldsymbol{x}}_i, \boldsymbol{g}_{c_i}) &= \underbrace{\min}_{k=1,\ldots,K} d_2^2(\tilde{\boldsymbol{x}}_i, \boldsymbol{g}_k) \\
d_2^2(\tilde{\boldsymbol{x}}_i, \boldsymbol{g}_{c_i}) &= \underbrace{\min}_{k=1,\ldots,K} \sum_{j=1}^{p} \sum_{z=1}^{Z_j} (x_{iz}^j - w_{kz}^j)^2
\end{aligned}
$$

Mise à jour des vecteurs prototypes. La distance euclidienne étant utilisée pour la recherche du neurone vainqueur, chaque vecteur prototype vaut la moyenne pondérée des observations qui appartiennent à sa région de Voronoï (les poids étant les valeurs de la fonction de voisinage). Donc, la mise à jour des vecteurs prototypes se fait de la manière suivante :

$$w_{kz}^{j}(t+1) = \frac{\sum_{i=1}^{n} h_{kc_i}(t) x_{iz}^{j}}{\sum_{i=1}^{n} h_{kc_i}(t)}$$

où $k \in \{1, \ldots, K\}$, $j \in \{1, \ldots, p\}$, $z \in \{1, \ldots, Z_j\}$ et $h_{kc_i}(t)$ est la fonction de voisinage gaussienne (voir équation (1.19), p.27).

À la fin de l'algorithme, les données seront classifiées en K classes. Chaque classe k contiendra toutes les observations ayant le neurone k comme neurone vainqueur. Outre cette tâche de classification, le fait d'utiliser les valeurs de la fonction de voisinage comme poids dans la formule de mise à jour des vecteurs prototypes, garantit la préservation de la topologie. L'algorithme 2.7 présente les étapes de cette méthode.

Algorithme 2.7 Apprentissage en mode différé des cartes auto-organisatrices pour des données symboliques homogénéisées.

Entrées : Fournir :

- $\boldsymbol{X}_i$ {□ Les observations décrites par des variables symboliques mixtes}
- lig, col {□ Nombre de lignes et de colonnes de la carte $K = lig \cdot col$}
- T {□ Nombre total d'itérations}
- σ_{init}, σ_{final} {□ Valeurs initiale et finale du rayon de voisinage}

Sorties : Récupérer :

- $\boldsymbol{g}_k(T)$, $k \in \{1, \ldots, K\}$ {□ les vecteurs prototypes auto-organisés}
- $P_T \leftarrow \{C_1, \ldots, C_K\}$ {□ La partition finale.}

Homogénéisation des données :

Pour $i = 1 \rightarrow n$ **Faire**

Transformer $\boldsymbol{X}_i$ en vecteur d'histogrammes $\tilde{\boldsymbol{x}}_i$ (voir sous-section 2.5.1, p.68)

Fin pour

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Initialiser les vecteurs prototypes $\boldsymbol{g}_k(0)$, $k \in \{1, \ldots, K\}$

Apprentissage :

Répéter

$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$

Calculer les neurones vainqueurs des observations - génération de la partition P_t :

Pour $i = 1 \rightarrow n$ **Faire**

$d_2^2(\tilde{\boldsymbol{x}}_i, \boldsymbol{g}_{c_i}) \leftarrow \underbrace{\min}_{k=1,\ldots,K} \sum_{j=1}^{p} \sum_{z=1}^{Z_j} (x_{iz}^j - w_{kz}^j)^2$

$C_{c_i} \leftarrow \tilde{\boldsymbol{x}}_i$ {□ Affecter l'individu $\tilde{\boldsymbol{x}}_i$ à la classe C_{c_i}}

Fin pour

Calculer les valeurs de la fonction de voisinage :

Pour $i = 1 \rightarrow n$ **Faire**

Pour $k = 1 \rightarrow K$ **Faire**

$h_{kc_i}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_i} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Mettre à jour des vecteurs prototypes $\boldsymbol{g}_k(t)$ *:*

Pour $k = 1 \rightarrow K$ **Faire**

Pour $j = 1 \rightarrow p$ **Faire**

Pour $z = 1 \rightarrow Z_j$ **Faire**

$w_{kz}^j(t+1) \leftarrow \frac{\sum_{i=1}^{n} h_{kc_i}(t) x_{iz}^j}{\sum_{i=1}^{n} h_{kc_i}(t)}$

Fin pour

Fin pour

Fin pour

$t \leftarrow t + 1$

Jusqu'à $t > T$

Retourner $\boldsymbol{g}_k(T)$, $k \in \{1, \ldots, K\}$ et $P_T = \{C_1, \ldots, C_K\}$

Visualisation de la carte

Pour les ensembles de données de grande dimension et dans le but de pouvoir visualiser la carte, nous proposons de projeter les vecteurs de données et les prototypes des neurones dans un espace bidimensionnel en utilisant la technique mise en place par Makosso K. (2010) qui consiste à transformer les histogrammes en intervalles, pour ensuite pouvoir appliquer l'analyse en composantes principales pour les données de type intervalle (Cazes *et al.*, 1997) (voir annexe B, p.189). Pour chaque histogramme $\tilde{x}_i^j = (U_j; \boldsymbol{x}_i^j)$, un score l_z^j est affecté à chaque modalité f_z^j qui est le rang de cette dernière dans son ensemble. Par exemple, f_1^j a un score 1, f_2^j a un score 2 et $f_{Z_j}^j$ a un score Z_j. La moyenne m_i^j de chaque histogramme est calculée selon :

$$m_i^j = \sum_{z=1}^{z_j} l_z^j x_{iz}^j$$

Chaque histogramme $\tilde{x}_i^j$ est transformé en un intervalle $[a_i^j, b_i^j]$ en utilisant l'inégalité de Tchebychev :

Soit X l'ensemble de valeurs formées par les composantes du vecteur $\boldsymbol{x}_i^j$ et soit m_i^j la moyenne empirique de X et δ_i^j l'écart-type empirique de X. Selon Tchebychev :

$$P(X \in [a_i^j = m_i^j - \eta\delta_i^j, b_i^j = m_i^j + \eta\delta_i^j]) > 1 - \frac{1}{\eta^2}$$

Nous prenons pour η la valeur 2.

2.5.2 2ème approche : Carte auto-organisatrice pour des données mixtes

Cette approche ne requiert aucune transformation de données. Elle est fondée sur l'algorithme *DissSOM* (voir sous-section 2.4.3, p.65), où l'entraînement de la carte se fait en utilisant les dissimilarités, ou les distances entre les données. La matrice des distances est construite en utilisant la distance de Minkowski généralisée (voir sous-section 2.3.2, p.55) qui permet de calculer la proximité entre deux individus décrits par des variables symboliques mixtes. L'algorithme d'apprentissage que nous proposons est le même que l'algorithme *DissSOM*, mais nous supposons qu'un vecteur prototype, ou un individu référent, est une observation de l'ensemble Ω ($\ell = 1$). Les vecteurs prototypes initiaux sont choisis aléatoirement dans l'ensemble des observations. L'algorithme d'apprentissage de la carte se résume en deux phases : la phase **Affectation** où se fait la recherche du neurone vainqueur de chaque observation, et la phase **Représentation** dans laquelle se fait le calcul, ou la détermination des nouveaux vecteurs prototypes. L'algorithme 2.8 détaille les étapes de la méthode proposée. À la fin de l'algorithme, le vecteur prototype de chaque classe est l'observation dont la somme des distances aux autres observations est minimale.

Algorithme 2.8 Apprentissage par batch des cartes auto-organisatrices pour des données symboliques mixtes.

Entrées : Fournir :

- $\boldsymbol{X}_i$ {□ Les observations décrites par des variables symboliques mixtes}

- lig, col {□ Dimensions de la carte. $K = lig \cdot col$}
- T {□ Nombre total d'itérations}
- σ_{init}, σ_{final} {□ Valeurs initiale et finale du rayon de voisinage}

Sorties : Récupérer :

- $\boldsymbol{W}_k(T)$, $k \in \{1, \ldots, K\}$ {□ les vecteurs prototypes auto-organisés}
- $P_T \leftarrow \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

Calculer la matrice des distances D

Pour $i = 1 \rightarrow n$ **Faire**

Pour $i' = 1 \rightarrow n$ **Faire**

Calculer $D(i, i') \leftarrow d\big(\boldsymbol{X}_i, \boldsymbol{X}_{i'}\big)$ {□ d : distance de Minkowski généralisée (équation (2.3)}

Fin pour

Fin pour

$t \leftarrow 0$ {□ t : Itération courante}

Choisir les vecteurs prototypes initiaux $\boldsymbol{W}_k$, $k \in \{1, \ldots, K\}$ parmi les observations

Répéter

Apprentissage :

$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$

Calculer les neurones vainqueurs des observations, générer une partition P_t :

Pour $i = 1 \rightarrow n$ **Faire**

$c_i = \arg\min_k \sum_{r=1}^{K} h_{kr}(t) d^2\big(\boldsymbol{X}_i, \boldsymbol{W}_r\big)$

$C_{c_i} \leftarrow \boldsymbol{X}_i$ {□ Affecter l'individu $\boldsymbol{X}_i$ à la classe C_{c_i}}

Fin pour

Calculer les valeurs de la fonction de voisinage $h_{kc_i}(t)$

Pour $i = 1 \rightarrow n$ **Faire**

Pour $k = 1 \rightarrow K$ **Faire**

$h_{kc_i}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_i} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Déterminer les vecteurs prototypes

Pour $k = 1 \rightarrow K$ **Faire**

$\upsilon \leftarrow \arg\min_{i'} \sum_{i=1}^{n} h_{kc_i}(t) d^2\big(\boldsymbol{X}_{i'}, \boldsymbol{X}_i\big)$

$\boldsymbol{W}_k(t+1) \leftarrow \boldsymbol{X}_\upsilon$

Fin pour

$t \leftarrow t + 1$ {□ Itération courante}
Jusqu'à $t > T$
Retourner $\boldsymbol{W}_k(T)$, $k \in \{1, \ldots, K\}$ et $P_T = \{C_1, \ldots, C_K\}$

2.6 Étude expérimentale

Dans le but de valider les deux méthodes proposées, nous les avons utilisées pour classifier deux jeux de données symboliques. Le premier est un jeu de données symboliques mixtes et le second est un jeu de données de type intervalle. Nous comparons nos méthodes à la méthode *S-SOM* (Yang *et al.*, 2012) qui consiste à créer une carte auto-organisatrice pour les données symboliques mixtes en adoptant la même structure pour les neurones que celle proposée par El-Sonbaty et Ismail (1998), où chaque neurone est représenté par un ensemble flou formé de toutes les valeurs possibles que prennent les p variables. Dans la méthode *S-SOM*, à chaque valeur est associé un degré d'appartenance de cette valeur à la variable correspondant et l'apprentissage de la carte auto-organisatrice est réalisé en mettant à jour ces degrés d'appartenance et en utilisant la dissimilarité modifiée de Gowda et Diday (voir équations (2.4), p.60) pour la recherche du neurone vainqueur d'une observation symbolique.

2.6.1 Jeu de données Huiles

Ce jeu de données regroupe des informations concernant des huiles et des graisses (Ichino et Yaguchi, 1994). Il contient $n = 8$ observations, chacune décrite par 4 variables de type intervalle et une variable multivaluée (voir tableau 2.6, p.77). Il est connu que les paires d'observations (1,2) ;(3,4) ;(5,6) et (7,8) ont des propriétés similaires. La figure 2.2 représente le dendrogramme résultant de la classification hiérarchique ascendante de ce jeu de données en utilisant le critère du saut minimal (Ichino et Yaguchi, 1994), sachant que la distance euclidienne généralisée est utilisée pour calculer la proximité entre deux observations (voir équation (2.3), p.59) avec une valeur pour γ valant 0.5 et une valeur commune pour les poids ζ_j, $j \in \{1, \ldots, 5\}$ qui vaut 1/5 (en supposant que toutes les variables sont de même importance).

Tableau 2.6 – Jeu de données Huiles.

Observation	Densité	Pt de congélation	Ind. d'Iode	Ind. de Saponification	Acides gras
1-Huile de lin	[0.930,0.935]	[-27,-18]	[170,204]	[118,196]	{L, Ln, O, P, M}
2-Huile de perilla	[0.930,0.937]	[-5,-4]	[192,208]	[188,197]	{L, Ln, O, P, S}
3-Huile de coton	[0.916,0.918]	[-6,-1]	[99,113]	[189,198]	{L, O, P, M, S}
4-Huile de sésame	[0.920,0.926]	[-6,-4]	[104,116]	[187,193]	{L, O, P, S, A}
5-Huile de camelia	[0.916,0.917]	[-21,-15]	[80,82]	[189,193]	{L, O}
6-Huile d'olive	[0.914,0.919]	[0,6]	[79,90]	[187,196]	{L, O, P, S}
7-Suif de boeuf	[0.860,0.870]	[30,38]	[40,48]	[190,199]	{O, P, M, S, C}
8-Graisse de porc	[0.858,0.864]	[22-32]	[53,77]	[190,202]	{L, O, P, M, S, Lu}

L : acide linoléique, Ln : acide linolénique, O : acide oléique, P : acide palmitique, M : acide myristique, S : acide stéarique, A : acide arachidique, C : acide caprique, Lu : acide laurique

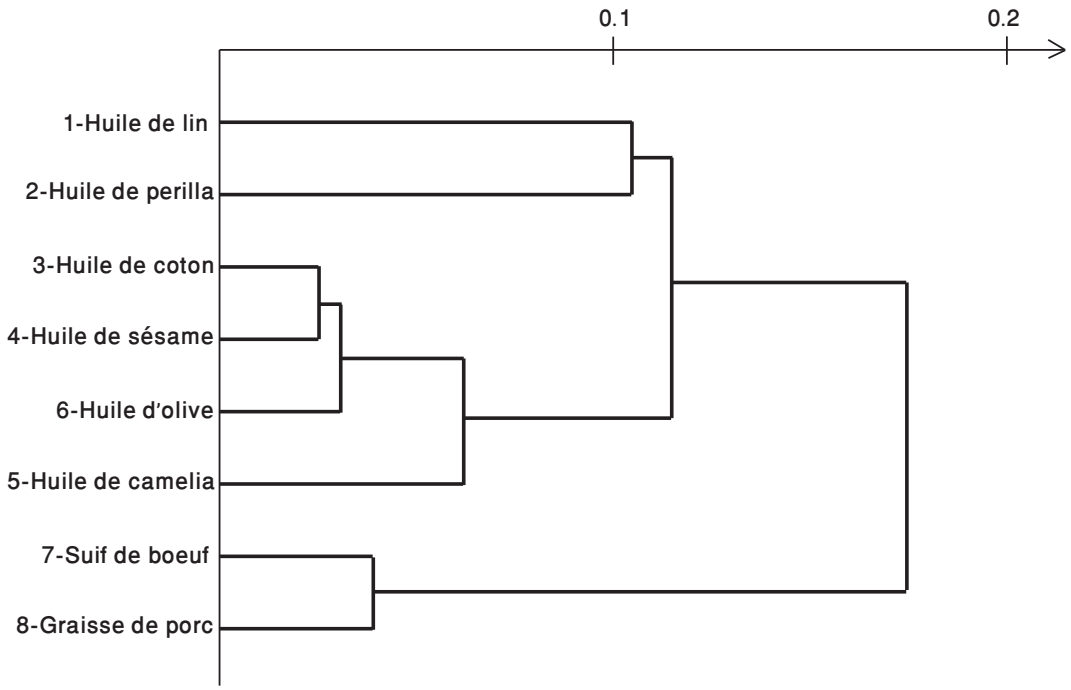


FIGURE 2.2 – Dendrogramme avec le critère du saut minimal pour le jeu de données Huiles.

Carte auto-organisatrice pour des données symboliques homogénéisées (1ère approche). Après avoir homogénéisé les données suivant la technique décrite à la sous-section 2.5.1, les données résultantes sont utilisées pour entraîner une carte auto-organisatrice unidimensionnelle de $K = 3$ neurones ($lig = 1$, $col = 3$).

L'apprentissage de la carte est fait suivant l'algorithme 2.7. Le rayon de voisinage a une valeur initiale de $\sigma_{init} = 1.5$ et une valeur finale de $\sigma_{final} = 0.1$. Le nombre total d'itérations est $T = 100$ et les vecteurs prototypes initiaux sont choisis aléatoirement parmi les observations.

Visualisation et qualité de la carte. Afin de pouvoir visualiser les données et les vecteurs prototypes, nous utilisons la technique décrite dans l'annexe B afin de projeter la carte et les données sur un espace bidimensionnel. La figure 2.3 montre le résultat de l'analyse en composantes principales des données et des vecteurs prototypes connectés par leur centre. Nous remarquons un bon degré de déploiement de la carte sur les données et un bon degré de préservation de la topologie (chaque prototype est connecté à ses voisins). L'erreur topographique définie dans l'équation (1.24) est de 0%.

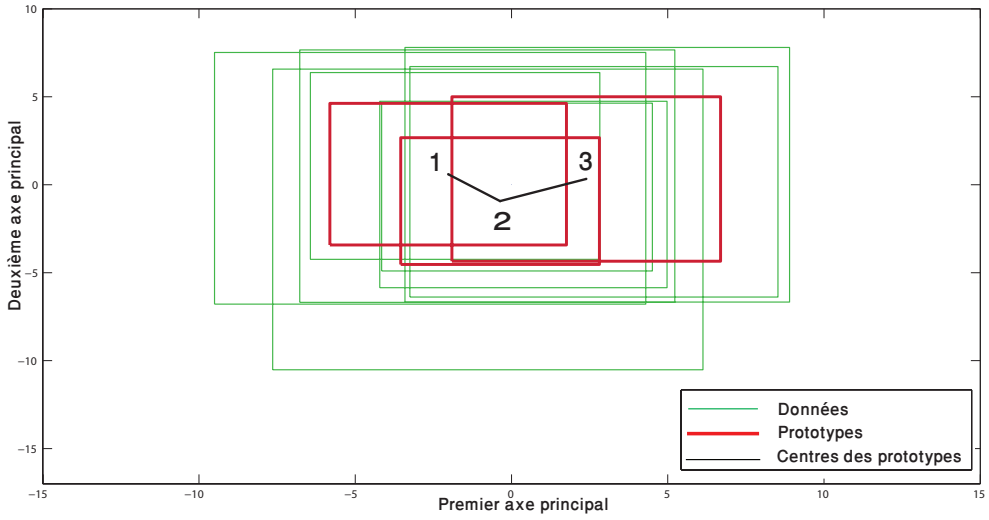


FIGURE 2.3 – Projection des données et des prototypes pour le jeu de données Huiles.

Résultats de la classification. En affectant chaque observation à son neurone vainqueur, nous obtenons une partition de 3 classes. La figure 2.4 montre la répartition des observations sur les trois neurones. En comparant ce résultat avec la méthode de la classification hiérarchique ascendante (Ichino et Yaguchi, 1994), nous obtenons des résultats similaires. En effet, en examinant le dendrogramme de la figure 2.2, nous remarquons qu'en coupant le dendrogramme de façon à avoir 3

classes, les observations 3, 4, 5 et 6 appartiennent à la même classe alors que dans notre approche, l'observation 4 appartient à la même classe que les deux premières observations. Ajoutons aussi qu'en traitant les données avec une carte auto-organisatrice, nous obtenons une information supplémentaire sur la proximité des classes. Les individus appartenant à la classe C_3 sont plus proches de ceux de la classe C_2 que ceux de la classe C_1.

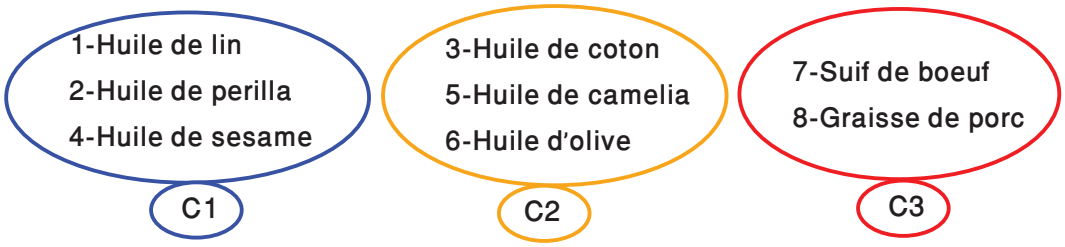


FIGURE 2.4 – Carte auto-organisatrice obtenue pour le jeu de données Huiles avec l'algorithme *SOM* pour données homogénéisées.

Cette même approche est utilisée dans Hajjar et Hamdan (2012c) et testée moyennant deux autres jeux de données symboliques réelles.

Carte auto-organisatrice pour des données symboliques mixtes (2ème approche). L'apprentissage d'une carte auto-organisatrice unidimensionnelle de $K = 3$ neurones est fait suivant l'algorithme 2.8 avec les mêmes paramètres pour la carte choisis dans la première méthode ($\sigma_{init} = 1.5$, $\sigma_{final} = 0.1$, $T = 100$). L'entrée de l'algorithme est la matrice de distances obtenue en calculant les distances euclidiennes généralisées entre les observations suivant l'équation (2.3), en affectant au paramètre γ la valeur 0.5 et aux poids ζ_j, $j \in \{1, \ldots, 5\}$ la même valeur 1/5 (en supposant que toutes les variables sont de même importance). Le tableau 2.7 représente la matrice des distances du jeu de données Huiles.

Tableau 2.7 – Matrice des distances pour le jeu de données Huiles.

	obs.1	**obs.2**	**obs.3**	**obs.4**	**obs.5**	**obs.6**	**obs.7**	**obs.8**
obs.1	0	0.1046	0.1493	0.1456	0.1632	0.1748	0.3136	0.2946
obs.2	0.1046	0	0.1217	0.1128	0.1575	0.1464	0.2839	0.2647
obs.3	0.1493	0.1217	0	0.0289	0.0634	0.0349	0.1913	0.1773
obs.4	0.1456	0.1128	0.0289	0	0.0646	0.0440	0.2103	0.1963
obs.5	0.1632	0.1575	0.0634	0.0646	0	0.0690	0.2184	0.2035
obs.6	0.1748	0.1464	0.0349	0.0440	0.0690	0	0.1720	0.1623
obs.7	0.3136	0.2839	0.1913	0.2103	0.2184	0.1720	0	0.0481
obs.8	0.2946	0.2647	0.1773	0.1963	0.2035	0.1623	0.0481	0

Résultats de la classification. La figure 2.5 montre la répartition des observations sur les neurones de la carte auto-organisatrice. Les vecteurs prototypes ou individus référents sont encadrés dans chaque classe. En calculant la somme des distances entre une observation donnée et les autres observations de la deuxième classe, nous constatons qu'avec l'observation 3, cette somme de distances est minimale. En effet :

- $d(3,4) + d(3,5) + d(3,6) = 0.1270$
- $d(4,3) + d(4,5) + d(4,6) = 0.1376$
- $d(5,3) + d(5,4) + d(5,6) = 0.1971$
- $d(6,3) + d(6,4) + d(6,5) = 0.1478$

Ce qui justifie que l'observation « 3-Huile de coton » est le vecteur prototype ou l'individu référent de la classe C_2. Le résultat de cette classification correspond parfaitement à celui obtenu quand la méthode de classification hiérarchique est utilisée (Ichino et Yaguchi, 1994) en coupant le dendrogramme de la figure 2.2 de façon à avoir 3 classes. En outre, d'après l'organisation des neurones sur la carte auto-organisatrice, nous pouvons conclure que les individus de la classe C_1 sont plus proches de ceux de la classe C_2 que ceux de la classe C_3, sachant que l'erreur topographique obtenue est $te = 0\%$ (voir équation (1.24), p.34). Il est à noter que la première approche (SOM pour données homogénéisées) donne des résultats comparables à cette approche sauf pour l'observation 4.

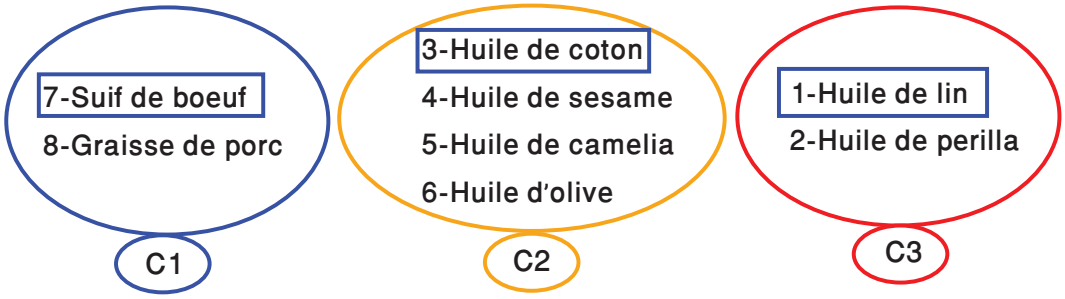


FIGURE 2.5 – Carte auto-organisatrice pour le jeu de données Huiles obtenue avec l'algorithme *SOM* pour données mixtes.

Comparaison avec d'autres méthodes. En guise de comparaison, le jeu de données Huiles est utilisé pour entraîner une carte auto-organisatrice de 3 neurones suivant l'algorithme *S-SOM* (Yang *et al.*, 2012). Le tableau 2.8 montre la répartition des observations sur les 3 classes, ce qui permet de conclure que le résultat obtenu avec la carte auto-organisatrice pour des données mixtes correspond le plus à celui obtenu dans (Ichino et Yaguchi, 1994).

Tableau 2.8 – Résultats de la classification pour le jeu de données Huiles avec l'algorithme *S-SOM*.

Classe	Observations
C_1	1-Huile de lin
C_2	2-Huile de perilla 3-Huile de coton 4-Huile de sésame 5-Huile de camelia 6-Huile d'olive
C_3	7-Suif de boeuf 8-Graisse de porc

2.6.2 Jeu de données Températures

Ce jeu de données contient uniquement des données de type intervalle. Il correspond aux températures minimales et maximales enregistrées durant les 12 mois de l'année dans 37 villes du monde entier (De Carvalho et De Souza, 2010). Le tableau 2.9 illustre ce jeu de données où les 37 observations sont décrites par 12 variables de type intervalle.

Carte auto-organisatrice pour des données symboliques homogénéisées (1ère approche). Après avoir homogénéisé les données suivant la technique décrite à la sous-section 2.5.1, les données résultantes sont utilisées pour entraîner une carte

Tableau 2.9 – Jeu de données Températures.

Obs.	Villes	Jan.	Fév.	Mar.	Avr.	Mai	Juin	Juil.	Août	Sep.	Oct.	Nov.	Déc.
1	Amsterdam	[-4,4]	[5,3]	[2,12]	[5,15]	[7,17]	[10,20]	[10,20]	[12,23]	[0,20]	[5,15]	[1,10]	[-1,4]
2	Athens	[6,12]	[6,12]	[8,16]	[11,19]	[16,25]	[19,29]	[22,32]	[22,32]	[19,28]	[16,23]	[11,18]	[8,14]
3	Bahrain	[13,19]	[14,19]	[17,23]	[21,27]	[25,32]	[28,34]	[29,36]	[30,36]	[28,34]	[24,31]	[20,26]	[15,21]
4	Bombay	[19,28]	[19,28]	[22,30]	[24,32]	[27,33]	[26,32]	[25,30]	[25,30]	[24,30]	[24,32]	[23,32]	[20,30]
5	Cairo	[8,20]	[9,22]	[11,25]	[14,29]	[17,33]	[20,35]	[22,36]	[22,35]	[20,33]	[18,31]	[14,26]	[10,20]
6	Calcutta	[13,27]	[16,29]	[21,34]	[24,36]	[26,36]	[26,33]	[26,32]	[26,32]	[26,32]	[24,32]	[18,29]	[13,26]
7	Colombo	[22,30]	[22,30]	[23,31]	[24,31]	[25,31]	[25,30]	[25,29]	[25,29]	[25,30]	[24,29]	[23,29]	[22,30]
8	Copenhagen	[-2,2]	[-3,2]	[-1,5]	[3,10]	[8,16]	[11,20]	[14,22]	[14,21]	[11,18]	[7,12]	[3,7]	[1,4]
9	Dubai	[13,23]	[14,24]	[17,28]	[19,31]	[22,34]	[25,36]	[28,39]	[28,39]	[25,37]	[21,34]	[17,30]	[14,26]
10	Frankfurt	[-10,9]	[-8,10]	[-4,17]	[0,24]	[3,27]	[7,30]	[8,32]	[8,31]	[5,27]	[0,22]	[-3,14]	[-8,10]
11	Geneva	[-3,5]	[-6,6]	[3,9]	[7,13]	[10,17]	[15,17]	[16,24]	[16,23]	[11,19]	[6,13]	[3,8]	[-2,6]
12	Hong Kong	[13,17]	[12,16]	[15,19]	[19,23]	[22,27]	[25,29]	[25,30]	[25,30]	[25,29]	[22,27]	[18,23]	[14,19]
13	KualaLumpur	[22,31]	[23,32]	[23,33]	[23,33]	[23,32]	[23,32]	[23,31]	[23,32]	[23,32]	[23,31]	[23,31]	[23,31]
14	Lisbon	[8,13]	[8,14]	[9,16]	[11,18]	[13,21]	[16,24]	[17,26]	[18,27]	[17,24]	[14,21]	[11,17]	[8,14]
15	London	[2,6]	[2,7]	[3,10]	[5,13]	[8,17]	[11,20]	[13,22]	[13,21]	[11,19]	[8,14]	[5,10]	[3,7]
16	Madras	[20,30]	[20,31]	[22,33]	[26,35]	[28,39]	[27,38]	[26,36]	[26,35]	[25,34]	[24,32]	[22,30]	[21,29]
17	Madrid	[1,9]	[1,12]	[3,16]	[6,19]	[9,24]	[13,29]	[16,34]	[16,33]	[13,28]	[8,20]	[4,14]	[1,9]
18	Manila	[21,27]	[22,27]	[24,29]	[24,31]	[25,31]	[25,31]	[23,29]	[24,28]	[25,28]	[24,29]	[22,28]	[22,27]
19	Mauritius	[22,28]	[22,29]	[22,29]	[21,28]	[19,25]	[18,24]	[17,23]	[17,23]	[17,24]	[18,25]	[19,27]	[21,28]
20	Mexico City	[6,22]	[15,23]	[17,25]	[18,27]	[18,27]	[18,27]	[18,27]	[18,26]	[18,26]	[16,25]	[14,25]	[8,23]
21	Moscow	[-13,-6]	[-12,-5]	[-8,0]	[0,8]	[7,18]	[11,23]	[13,24]	[11,22]	[6,16]	[1,8]	[-5,0]	[-11,5]
22	Munich	[-6,1]	[-5,3]	[-2,9]	[3,14]	[7,18]	[10,21]	[12,23]	[11,23]	[8,20]	[4,13]	[0,7]	[-4,2]
23	Nairobi	[12,25]	[13,26]	[14,25]	[14,24]	[13,22]	[12,21]	[11,21]	[11,21]	[11,24]	[13,24]	[13,23]	[13,23]
24	New Delhi	[6,21]	[10,24]	[14,29]	[20,36]	[26,40]	[28,39]	[27,35]	[26,34]	[24,34]	[18,34]	[11,28]	[7,23]
25	New York	[-2,4]	[-3,4]	[1,9]	[6,15]	[12,22]	[17,27]	[21,29]	[20,28]	[16,24]	[11,19]	[5,12]	[-2,6]
26	Paris	[1,7]	[1,7]	[2,12]	[5,16]	[8,19]	[12,22]	[14,24]	[13,24]	[11,21]	[7,16]	[4,10]	[1,6]
27	Rome	[4,11]	[5,13]	[7,16]	[10,19]	[13,23]	[17,28]	[20,31]	[20,31]	[17,27]	[13,21]	[9,16]	[5,12]
28	San Francisco	[6,13]	[6,14]	[7,17]	[8,18]	[10,19]	[11,21]	[12,22]	[12,22]	[12,23]	[11,22]	[8,18]	[6,14]
29	Seoul	[0,7]	[1,6]	[1,8]	[6,16]	[12,22]	[16,25]	[18,31]	[16,30]	[9,28]	[3,24]	[7,19]	[1,8]
30	Singapore	[23,30]	[23,30]	[24,31]	[24,31]	[24,30]	[25,30]	[25,30]	[25,30]	[24,30]	[24,30]	[24,30]	[23,30]
31	Stockholm	[-9,-5]	[-9,-6]	[-4,2]	[1,8]	[6,15]	[11,19]	[14,22]	[13,20]	[9,15]	[5,9]	[1,4]	[-2,2]
32	Sydney	[20,30]	[20,30]	[18,26]	[16,23]	[12,20]	[5,17]	[8,16]	[9,17]	[11,20]	[13,22]	[16,26]	[20,30]
33	Tehran	[0,5]	[5,8]	[10,15]	[15,18]	[20,25]	[28,30]	[36,38]	[38,40]	[28,30]	[18,20]	[9,12]	[-5,0]
34	Tokyo	[0,9]	[0,10]	[3,13]	[9,18]	[14,23]	[18,25]	[22,29]	[23,31]	[20,27]	[13,21]	[8,16]	[2,12]
35	Toronto	[-8,-1]	[-8,-1]	[-4,4]	[-2,11]	[-8,18]	[13,24]	[16,27]	[16,26]	[12,22]	[6,14]	[-1,17]	[-5,1]
36	Vienna	[-2,1]	[-1,3]	[1,8]	[5,14]	[10,19]	[13,22]	[15,24]	[14,23]	[11,19]	[7,13]	[2,7]	[1,3]
37	Zurich	[-11,9]	[-8,15]	[-7,18]	[-1,21]	[2,27]	[6,30]	[10,31]	[8,25]	[5,23]	[3,22]	[0,19]	[-11,8]

auto-organisatrice de $K = 4$ neurones arrangés suivant une grille carrée ($lig = 2$, $col = 2$). L'apprentissage de la carte est fait suivant l'algorithme 2.7. Le rayon de voisinage a une valeur initiale de $\sigma_{init} = 1$ et une valeur finale de $\sigma_{final} = 0.1$. Le nombre total d'itérations est $T = 100$ et les vecteurs prototypes initiaux sont choisis aléatoirement parmi les observations.

Visualisation et qualité de la carte. Afin de pouvoir visualiser les données et les vecteurs prototypes, nous utilisons la technique décrite dans l'annexe B afin de projeter la carte sur un espace bidimensionnel. La figure 2.6 montre le résultat de l'analyse en composantes principales des données et des vecteurs prototypes connectés par leur centre. Nous remarquons un bon degré de déploiement de la carte sur les données et un bon degré de préservation de la topologie (Chaque prototype est connecté à ses voisins). L'erreur topographique (voir équation (1.24), p.34) est de 10.81%.

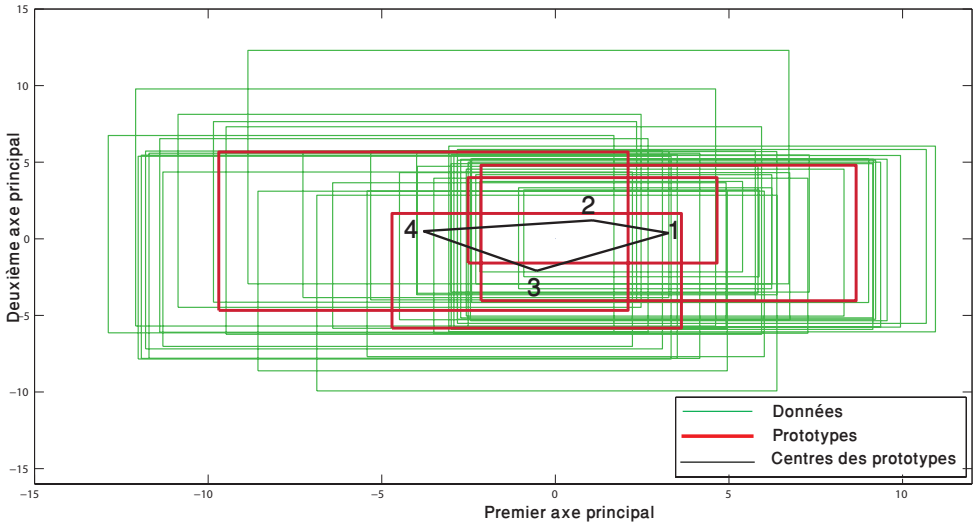


FIGURE 2.6 – Projection des données et des prototypes pour le jeu de données Températures.

Résultats de la classification. En affectant chaque observation à son neurone vainqueur, nous obtenons une partition de 4 classes. La figure 2.7 montre la répartition des observations sur les quatre neurones. Nous remarquons que les villes proches géographiquement ou situées à peu près sur la même latitude, sont affec-

tées au même neurone ou à un neurone voisin. Les deux neurones voisins C_2 et C_4 contiennent les villes les plus froides alors que les neurones C_1 et C_3 contiennent les villes les plus chaudes. Notons aussi qu'en se déplaçant vers le bas et vers la gauche sur la carte auto-organisatrice, nous retrouvons des villes plus froides. La figure 2.8 représente la carte du monde contenant les 37 villes où celles appartenant à la même classe sont reliées ensemble et dessinées avec la même couleur.

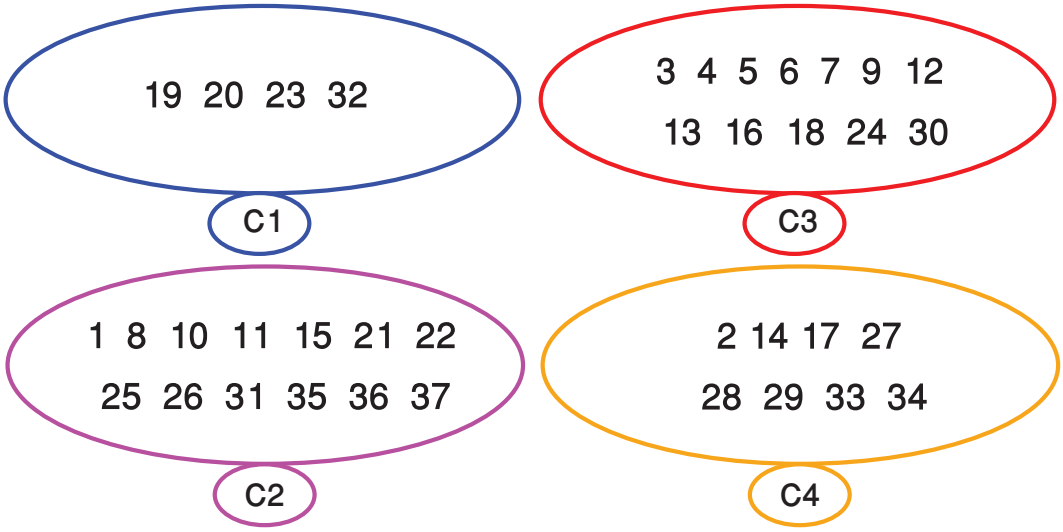


FIGURE 2.7 – Carte auto-organisatrice obtenue pour le jeu de données Températures avec l'algorithme *SOM* pour données homogénéisées.

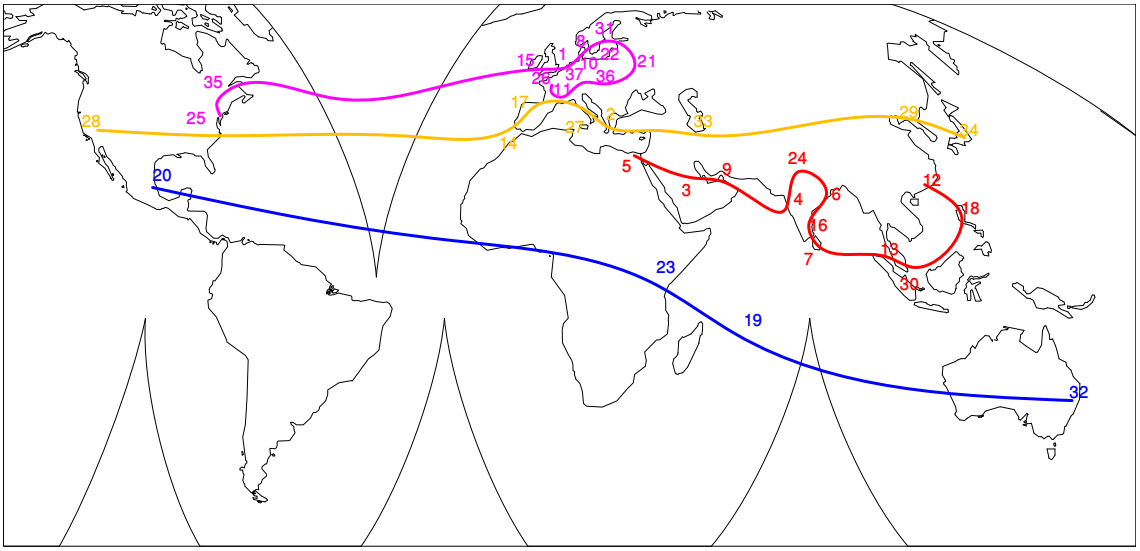


FIGURE 2.8 – Répartition des classes sur la carte du monde avec l'algorithme *SOM* pour données homogénéisées.

Carte auto-organisatrice pour des données symboliques mixtes (2ème approche). L'apprentissage d'une carte auto-organisatrice carrée de $K = 4$ neurones est fait suivant l'algorithme 2.8 avec les mêmes paramètres pour la carte choisis dans la première approche ($\sigma_{init} = 1$, $\sigma_{final} = 0.1$, $T = 100$). L'entrée de l'algorithme est la matrice de dissimilarités obtenue en calculant les distances euclidiennes généralisées entre les observations suivant l'équation (2.3), en affectant au paramètre γ la valeur 0.5 et aux poids ζ_j, $j \in \{1, \ldots, 12\}$ la même valeur $1/12$ (en supposant que toutes les variables soient de même importance). La figure 2.9 montre la répartition des villes sur les neurones de la carte auto-organisatrice. Les vecteurs prototypes ou individus référents sont encadrés dans chaque classe. Nous remarquons aussi qu'avec cette approche que les villes ayant un climat similaire sont affectées au même neurone ou à un neurone voisin sur la carte. L'erreur topographique obtenue est $te = 8.11\%$. La figure 2.10 représente la carte du monde où les villes appartenant à la même classe sont reliées ensemble et dessinées avec la même couleur.

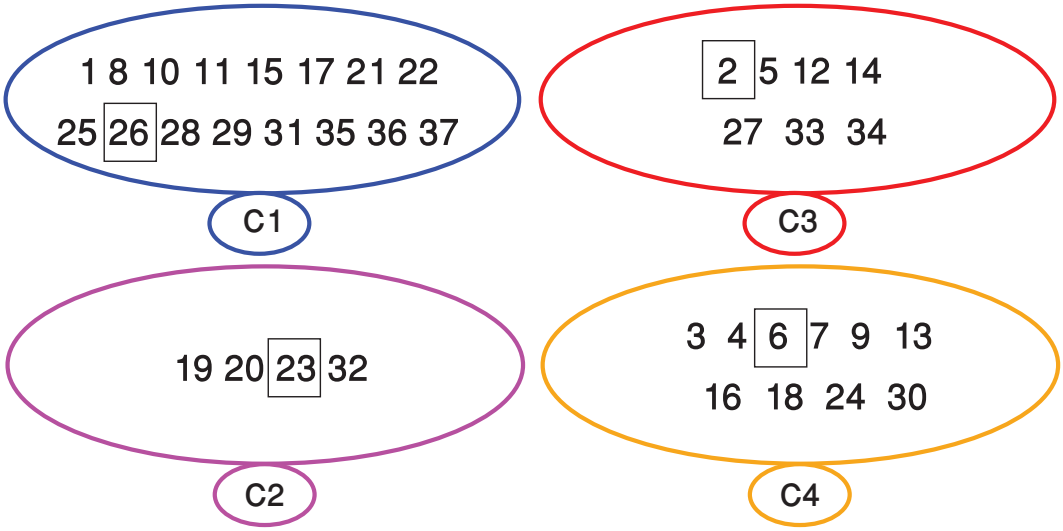


FIGURE 2.9 – Carte auto-organisatrice obtenue pour le jeu de données Températures avec l'algorithme *SOM* pour données mixtes.

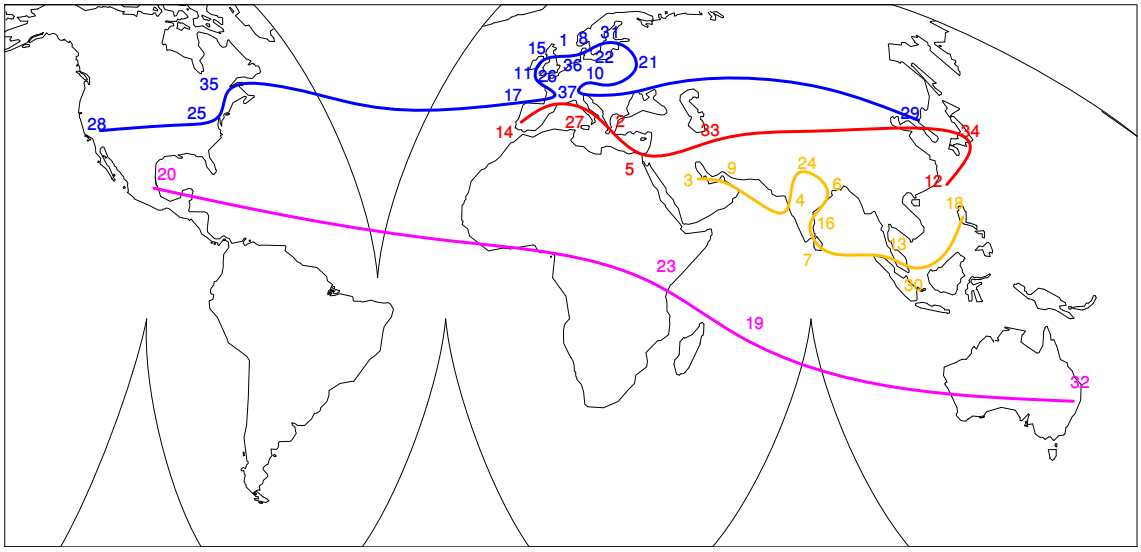


FIGURE 2.10 – Répartition des classes sur la carte du monde avec l'algorithme *SOM* pour données mixtes.

Comparaison avec d'autres méthodes. En guise de comparaison, le jeu de données Températures est utilisé pour entraîner une carte auto-organisatrice de 4 neurones suivant l'algorithme *S-SOM* (Yang *et al.*, 2012). Le tableau 3.8 montre la répartition des villes sur les 4 classes.

Tableau 2.10 – Résultats de la classification pour le jeu de données Températures avec l'algorithme *S-SOM*.

Classe	**Observations**
C_1	1 8 10 11 14 15 17 21 22 25 26 27 28 29 31 34 35 36 37
C_2	3 4 6 7 9 12 13 16 18 24 30
C_3	5 19 20 23 32
C_4	2 33

Nous avons calculé l'écart-type des latitudes des villes pour chaque classe (voir tableau 2.11, p.87). La méthode *SOM* pour données symboliques homogénéisées donne la plus petite somme d'écart-types.

Tableau 2.11 – Écart-type des latitudes par classe.

Classes	***SOM* pour données homogénéisées**	**SOM pour données mixtes**	***S-SOM* (Yang *et al.*, 2012)**
C_1	23.05	6.48	6.88
C_2	5.13	23.05	9.15
C_3	9.58	6.53	26.53
C_4	2.14	9.41	1.62
Total	39.90	45.47	44.18

2.7 Conclusion

Dans ce chapitre, nous avons évoqué les méthodes de classification existantes pour les données symboliques mixtes, après avoir passé en revue les distances et les dissimilarités proposées dans la littérature pour comparer deux observations décrites par des variables symboliques de tous types. Ensuite, nous avons mis en place deux méthodes de classification de données symboliques fondées sur l'apprentissage en mode différé des cartes auto-organisatrices. Dans la première méthode, une technique d'homogénéisation des données est requise pour les transformer en données de type histogramme qui seront présentées à la carte. Dans la deuxième méthode, une matrice de distances est construite en utilisant la distance de Minkowski généralisée, et l'apprentissage de la carte se fait alors avec les distances plutôt qu'avec les individus. Les deux méthodes ont été testées et comparées à une autre méthode en utilisant deux jeux de données symboliques. Bien que les résultats obtenus soient encourageants, il est nécessaire de préciser que la technique d'homogénéisation em-

ployée dans la première méthode risque de provoquer des distorsions dans les données, alors que la construction d'une matrice de distances dans la deuxième méthode risque d'être irréalisable pour les grands jeux de données faute d'espace mémoire.

Ce chapitre était consacré aux données symboliques où chaque observation est décrite par des variables de différents types. Un cas particulier intéressant des données symboliques est le cas des données de type intervalle où chaque observation est décrite exclusivement par des intervalles. Ce type de données a sollicité l'intérêt d'un grand nombre de chercheurs, et un bon nombre de travaux de recherche sont conduits sur l'analyse et la classification de données de type intervalle. Le chapitre 3 sera consacré aux méthodes de classification de données intervalles.

Chapitre 3

Cartes auto-organisatrices pour les données de type intervalle

3.1 Introduction

La représentation ponctuelle de phénomènes observés (naturels ou simulés) est limitée. Grâce aux récents développements des méthodes d'analyse de données, et des progrès technologiques, il est désormais possible de représenter les phénomènes sous diverses formes et d'envisager la représentation symbolique des données. La représentation de données sous formes d'intervalles est un cas particulier récurrent. Dans maintes applications, il est plus approprié d'utiliser les intervalles à la place de valeurs uniques. En fait, les intervalles permettent de prendre en considération la variation et la variabilité inhérentes aux données. Ils sont utilisés pour modéliser l'incertitude des résultats (observés et simulés). Dans de nombreux domaines, le stockage de données sous forme d'intervalles est ainsi utilisé pour résumer cette variabilité de l'information. Les applications météorologiques, financières, d'analyse comportementale, de contrôle de qualité sont des domaines usuels pour une telle représentation des données.

Les données de type intervalle ont sollicité l'intérêt de plusieurs chercheurs dans le domaine de l'analyse de données. Parmi ces travaux de recherche, on trouve : l'analyse en composantes principales (Cazes *et al.*, 1997; Gioia et C.N., 2006), l'analyse factorielle (Chouakria, 1998), la régression (Billard et Diday, 2000), le perceptron multicouche (Rossi et Conan-Guez, 2002) et la classification (voir sections 3.5, p.96

et 3.6, p.99).

Ce chapitre est consacré aux méthodes de classification de données intervalles. Nous donnons au début une définition des données intervalles et des distances existantes pour comparer deux vecteurs d'individus décrits par des variables de type intervalle. Ensuite, nous passons en revue les méthodes de classification existantes pour les données intervalles. Au cœur de ce chapitre, nous proposons de construire des cartes auto-organisatrices pour classifier les données intervalles en utilisant plusieurs distances, pour finir avec une étude expérimentale qui permet de mettre en valeur les méthodes proposées et de les comparer à des méthodes existantes.

3.2 Données de type intervalle et notations

Soit Ω un ensemble de n observations ou individus $\boldsymbol{\mathcal{R}}_i$, $i \in \{1, \ldots, n\}$. Chaque individu $\boldsymbol{\mathcal{R}}_i$ est décrit par p intervalles $\mathcal{R}_i^j = [a_i^j, b_i^j]$, $j \in \{1, \ldots, p\}$. Une variable $\mathcal{R}^j$ est une application de l'ensemble $\mathbb{I}^p$ (où $\mathbb{I} = \{[a, b];\ a \in \mathbb{R},\ b \in \mathbb{R},\ a \leqslant b\}$) dans U_j qui, à chaque observation $\boldsymbol{\mathcal{R}}_i$, fait correspondre la valeur prise par la variable $\mathcal{R}^j$ pour cette observation.

$$\begin{aligned} \mathcal{R}^j : \quad \mathbb{I}^p &\longrightarrow U_j \\ \boldsymbol{\mathcal{R}}_i &\longrightarrow \mathcal{R}^j(\boldsymbol{\mathcal{R}}_i) = \mathcal{R}_i^j = [a_i^j, b_i^j] \end{aligned}$$

Une observation $\boldsymbol{\mathcal{R}}_i$ est alors définie par un vecteur d'intervalles :

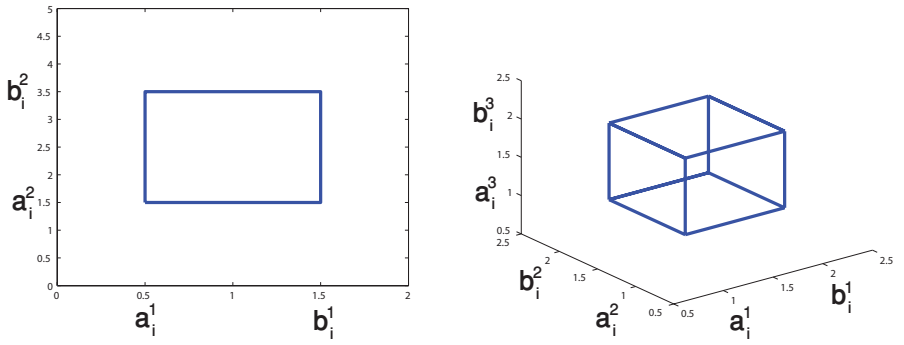
$$\boldsymbol{\mathcal{R}}_i = \left([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p]\right)^T$$

Donc, l'ensemble des observations est décrit par la matrice illustrée dans le tableau 3.1.

Tableau 3.1 – Matrice des observations décrites par des intervalles.

Obs.	$\mathcal{R}^1$	$\mathcal{R}^2$	...	$\mathcal{R}^p$
1	$[a_1^1, b_1^1]$	$[a_1^2, b_1^2]$	...	$[a_1^p, b_1^p]$
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
i	$[a_i^1, b_i^1]$	$[a_i^2, b_i^2]$	...	$[a_i^p, b_i^p]$
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
n	$[a_n^1, b_n^1]$	$[a_n^2, b_n^2]$	...	$[a_n^p, b_n^p]$

Une observation, décrite par p intervalles, est représentée graphiquement par un hyperrectangle (rectangle p-dimensionnel). Les figures 3.1 (a) et (b) montrent la représentation graphique d'une observation en deux dimensions ($p = 2$) et en trois dimensions ($p = 3$) respectivement.



(a) Deux dimensions (b) Trois dimensions

FIGURE 3.1 – Représentation graphique d'une observation décrite par des variables de type intervalle.

3.3 Mesures de proximité entre deux vecteurs d'intervalles

Dans ce qui suit, nous allons présenter les mesures de proximités existantes pour comparer deux vecteurs d'intervalles, entre autres : la distance L_2, la distance L_1, les distances basées sur la distance de Hausdorff, la distance *vertex-type* et les distances basées sur la distance de Mahalanobis.

3.3.1 Distance L_2

La dissemblance entre deux observations $\boldsymbol{\mathcal{R}}_i$ et $\boldsymbol{\mathcal{R}}_{i'}$ décrites par p variables de type intervalle est calculée en utilisant une extension de la distance de Minkowski de type L_2 aux données intervalles. Cette distance est définie par :

$$D_{L_2}(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) = \sqrt{\sum_{j=1}^{p} \left((a_i^j - a_{i'}^j)^2 + (b_i^j - b_{i'}^j)^2 \right)} \tag{3.1}$$

Le carré de la distance L_2 est une dissimilarité souvent utilisée pour calculer la dissemblance entre deux vecteurs d'intervalles. Elle est définie par :

$$D_{L_2}^2(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) = \sum_{j=1}^{p} \left((a_i^j - a_{i'}^j)^2 + (b_i^j - b_{i'}^j)^2 \right) \tag{3.2}$$

$D_{L_2}^2$, comme elle est définie, ne vérifie pas la propriété de l'inégalité triangulaire d'une distance (voir définition 1.2, p.13). Pour cela, on se contente de l'appeler une dissimilarité.

3.3.2 Distance L_1

La dissemblance entre deux observations $\boldsymbol{\mathcal{R}}_i$ et $\boldsymbol{\mathcal{R}}_{i'}$ décrites par p variables de type intervalle est calculée en utilisant une extension de la distance de Minkowski de type L_1 aux données intervalles. Cette distance est définie par :

$$D_{L_1}(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) = \sum_{j=1}^{p} \left(\left| a_i^j - a_{i'}^j \right| + \left| b_i^j - b_{i'}^j \right| \right) \tag{3.3}$$

D_{L_1}, comme elle est définie, vérifie toutes les propriétés d'une distance (voir définition 1.3, p.13).

3.3.3 Distances basées sur la distance de Hausdorff

La distance de Hausdorff est initialement définie pour comparer deux ensembles A et B. Tout d'abord, nous devons définir une fonction de comparaison entre un point a de l'ensemble A et l'ensemble B par :

$$h(a, B) = \inf_{b \in B} d(a, b)$$

La fonction de comparaison entre deux ensembles A et B est définie par :

$$h(A, B) = \sup_{a \in A} h(a, B)$$

En utilisant la norme L_1 pour calculer la distance entre a et b, $d(a,b) = |a-b|$, la distance entre deux ensembles A et B devient :

$$h(A,B) = \sup_{a\in A} \inf_{b\in B} |a-b|$$

où sup et inf sont respectivement le supremum et l'infimum d'un ensemble. Exemples : $h(3,\{4,5,7\}) = 1$ et $h(\{1,3\},\{4,5,7\}) = 3$.
$h(A,B)$ ne vérifie pas la propriété de symétrie d'une distance : $h(A,B) \neq h(B,A)$, par exemple :
$h(\{1,3\},\{4,5,7\}) = 3$ et $h(\{4,5,7\},\{1,3\}) = 4$. De plus $h(A,B) = 0 \not\Leftrightarrow A = B$. Pourtant, il est possible de définir une distance D_H entre A et B, appelée distance de Hausdorff, définie par :

$$d_H(A,B) = \max\{h(A,B), h(B,A)\}$$

La distance de Hausdorff entre deux intervalles $A_1 = [a_1, b_1]$ et $A_2 = [a_2, b_2]$ devient

$$d_H(A_1,A_2) = \max\{h(A_1,A_2), h(A_2,A_1)\} = \max\left(|a_1-a_2|, |b_1-b_2|\right)$$

La dissemblance entre deux vecteurs d'intervalles est calculée en utilisant des combinaisons de distances de Hausdorff entre intervalles élémentaires, nous distinguons :

Combinaison de type L_1 de distances de Hausdorff (Hausdorff-L_1). La distance entre deux vecteurs d'intervalles $\boldsymbol{\mathcal{R}}_i$ et $\boldsymbol{\mathcal{R}}_{i'}$ est calculée en utilisant une combinaison de type L_1 de distances de Hausdorff définie par :

$$D_{\text{Hausdorff-}L_1}(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) = \sum_{j=1}^{p} \Big(\max\left(|a_i^j - a_{i'}^j|, |b_i^j - b_{i'}^j|\right)\Big) \tag{3.4}$$

Combinaison de type L_2 de distances de Hausdorff (Hausdorff-L_2). La distance entre deux vecteurs d'intervalles $\boldsymbol{\mathcal{R}}_i$ et $\boldsymbol{\mathcal{R}}_{i'}$ est calculée en utilisant une combinaison de type L_2 de distances de Hausdorff définie par :

$$D_{\text{Hausdorff-}L_2}(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) = \sqrt{\sum_{j=1}^{p} \Big(\max\left(|a_i^j - a_{i'}^j|, |b_i^j - b_{i'}^j|\right)\Big)^2} \tag{3.5}$$

Combinaison de type L_∞ de distances de Hausdorff (Hausdorff-L_∞). La distance entre deux vecteurs d'intervalles $\boldsymbol{\mathcal{R}}_i$ et $\boldsymbol{\mathcal{R}}_{i'}$ est calculée en utilisant une combinaison de type L_∞ de distances de Hausdorff définie par :

$$D_{\text{Hausdorff-}L_\infty}(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) = \max_{j\in\{1,\ldots,p\}} \Big(\max \big(|a_i^j - a_{i'}^j|, |b_i^j - b_{i'}^j| \big) \Big) \tag{3.6}$$

3.3.4 Distance *vertex-type*

La distance *vertex-type* entre deux vecteurs d'intervalles, est la somme des carrés des distances euclidiennes entre les 2^p sommets des deux hyperrectangles correspondants. En désignant par $\boldsymbol{a_i} = (a_i^1, \ldots, a_i^p)^T$ le vecteur des coordonnées du sommet inférieur, et par $\boldsymbol{b_i} = (b_i^1, \ldots, b_i^p)^T$ le vecteur des coordonnées du sommet supérieur de l'hyperrectangle qui représente le vecteur $\boldsymbol{\mathcal{R}}_i$, cette distance se réduit à (Bock, 2008b) :

$$D_v(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) = 2^{p-1} \Big(\sum_{j=1}^{p} (a_i^j - a_{i'}^j)^2 + \sum_{j=1}^{p} (b_i^j - b_{i'}^j)^2 \Big) \tag{3.7}$$

3.3.5 Distance basée sur la distance de Mahalanobis

Nous rappelons que la distance de Mahalanobis permet de prendre en compte la corrélation entre les variables ainsi que leurs variabilités (voir définition 1.7, p.16). En calculant la somme de la distance de Mahalanobis entre les deux sommets inférieurs et les deux sommets supérieurs des deux hyperrectangles $\boldsymbol{\mathcal{R}}_i$ et $\boldsymbol{\mathcal{R}}_{i'}$ respectivement, nous obtenons :

$$\begin{aligned} D_M(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_{i'}) &= d_M(\boldsymbol{a}_i, \boldsymbol{a}_{i'}) + d_M(\boldsymbol{b}_i, \boldsymbol{b}_{i'}) \\ &= \sqrt{(\boldsymbol{a}_i - \boldsymbol{a}_{i'})^T \boldsymbol{S}^{-1} (\boldsymbol{a}_i - \boldsymbol{a}_{i'})} + \sqrt{(\boldsymbol{b}_i - \boldsymbol{b}_{i'})^T \boldsymbol{S}^{-1} (\boldsymbol{b}_i - \boldsymbol{b}_{i'})} \end{aligned} \tag{3.8}$$

où $\boldsymbol{a}_i$ et $\boldsymbol{b}_i$ sont respectivement le sommet inférieur et le sommet supérieur de l'hyperrectangle $\boldsymbol{\mathcal{R}}_i$ et $\boldsymbol{S}$ est la matrice de covariance intra-classe.

Autres distances. Il est possible aussi d'utiliser les distances définies entre vecteurs de données symboliques qui s'appliquent aux vecteurs d'intervalles et qui sont définies dans le chapitre 2, comme la distance de Ichino et Yaguchi (voir sous-section 2.3.2, p.55) ou la dissimilarité de Gowda et Diday (voir sous-section 2.3.1, p.52).

3.4 Normalisation de données de type intervalle

Dans le cas où les échelles de mesure des variables sont sensiblement différentes, il est recommandé de normaliser les données avant leur classification. La normalisation des variables de type intervalle se fait en calculant une mesure de dispersion pour les variables qui dépend de la distance utilisée (Chavent et Saracco, 2008). Nous évoquons dans ce qui suit, trois techniques de normalisation pour les variables de type intervalle correspondantes à la distance L_1, au carré de la distance L_2 et à la combinaison de type L_1 de distances de Hausdorff.

3.4.1 Distance L_1

Dans le cas où la distance entre vecteurs intervalles est la distance L_1, la mesure de dispersion $S^j_{L_1}$ pour la variable j est calculée comme suit :

$$S^j_{L_1} = \sum_{i=1}^{n} \left(|a^j_i - \alpha^j| + |b^j_i - \beta^j| \right)$$

$$\alpha^j = mediane(a^j_i)\,,\ i \in \{1, \ldots, n\}$$

$$\beta^j = mediane(b^j_i)\,,\ i \in \{1, \ldots, n\}$$

Chaque intervalle $[a^j_i, b^j_i]$ sera normalisé en $[\tilde{a}^j_i, \tilde{b}^j_i]$ tel que :

$$\tilde{a}^j_i = \frac{a^j_i}{S^j_{L_1}} \qquad \tilde{b}^j_i = \frac{b^j_i}{S^j_{L_1}}$$

3.4.2 Carré de la distance L_2

Dans le cas où la mesure de proximité entre vecteurs intervalles est le carré de la distance L_2, la mesure de dispersion $S^j_{L_2^2}$ pour la variable j est calculée ainsi :

$$S^j_{L_2^2} = \sum_{i=1}^{n} \left((a^j_i - \alpha^j)^2 + (b^j_i - \beta^j)^2 \right)$$

$$\alpha^j = \frac{\sum_{i=1}^{n} a^j_i}{n} \qquad \beta^j = \frac{\sum_{i=1}^{n} b^j_i}{n}$$

Chaque intervalle $[a^j_i, b^j_i]$ sera normalisé en $[\tilde{a}^j_i, \tilde{b}^j_i]$ tel que :

$$\tilde{a}^j_i = \frac{a^j_i}{\sqrt{S^j_{L_2^2}}} \qquad \tilde{b}^j_i = \frac{b^j_i}{\sqrt{S^j_{L_2^2}}}$$

3.4.3 Distance de Hausdorff-L_1

Dans le cas où la distance entre vecteurs d'intervalles est une combinaison de type L_1 de distances de Hausdorff, la mesure de dispersion $S^j_{\text{Hausdorff-}L_1}$ pour la variable j est calculée comme suit :

$$S^j_{\text{Hausdorff-}L_1} = \sum_{i=1}^{n} \left(|m_i^j - \alpha^j| + |l_i^j - \beta^j| \right)$$
$$\alpha^j = mediane(m_i^j) \ i \in \{1, \ldots, n\}$$
$$\beta^j = mediane(l_i^j) \ i \in \{1, \ldots, n\}$$

où $m_i^j = (a_i^j + b_i^j)/2$ et $l_i^j = (b_i^j - a_i^j)/2$

Chaque intervalle $[a_i^j, b_i^j]$ sera normalisé en $[\tilde{a}_i^j, \tilde{b}_i^j]$ tel que :

$$\tilde{a}_i^j = \frac{m_i^j - l_i^j}{S^j_{\text{Hausdorff-}L_1}} \quad \tilde{b}_i^j = \frac{m_i^j + l_i^j}{S^j_{\text{Hausdorff-}L_1}}$$

3.5 Méthodes de classification existantes pour les données intervalles

Plusieurs plateformes de recherche se sont concentrées sur la classification de données intervalles en travaillant sur l'extension des méthodes de classification existantes pour considérer ce type de données. Nous citons ci-après quelques unes de ces approches (voir figure 3.2).

Hamdan et Govaert ont développé une théorie sur la classification de données de type intervalle fondée sur l'estimation des **modèles de mélange**. Dans ce contexte, ils ont proposé deux approches d'estimation du maximum de vraisemblance à partir de données de type intervalle : l'approche mélange (Hamdan et Govaert, 2003a,b, 2005) et l'approche classification (Hamdan et Govaert, 2004a,c). L'approche mélange estime les paramètres du modèle à partir de données de type intervalle puis détermine la partition en rangeant chaque individu (un vecteur d'intervalles) dans la classe maximisant la probabilité d'appartenance *a posteriori* de l'individu calculée à partir des paramètres estimés. L'approche classification consiste à rechercher une partition de données de type intervalle de telle sorte que chaque classe soit assimilable à un sous-échantillon issue d'une composante du mélange. Il

s'agit donc d'estimer simultanément les paramètres du modèle et la partition recherchée à partir de données de type intervalle. L'intérêt de ces approches est de baser la classification de données de type intervalle sur les modèles de mélange qui, en plus de leurs autres avantages, sont bien adaptés aux cas de classes avec différents proportions, volumes, orientations et formes. Bock (2008a) a proposé un modèle probabiliste pour la classification de données symboliques de type intervalle.

L'algorithme des **nuées dynamiques** (voir sous-section 1.3.2, p.24) est largement utilisé dans la classification de données de type intervalle. Plusieurs méthodes basées sur l'algorithme des nuées dynamiques ont été développées. Elles se distinguent par le choix de la distance utilisée, ce qui entraîne une optimisation différente du critère des nuées dynamiques à chaque fois. Donc, la détermination des prototypes dépend de la distance choisie. Chavent et Lechevallier (2002) ont proposé un algorithme de nuées dynamiques en optimisant un critère basé sur une combinaison de type L_1 de distances de Hausdorff. Chavent (2004) a aussi proposé une méthode fondée sur les nuées dynamiques en utilisant comme distance entre vecteurs d'intervalles une combinaison de type L_∞ de distances de Hausdorff. De Souza et De Carvalho (2004) ont proposé deux méthodes de nuées dynamiques pour les données intervalles : la première méthode utilise la distance L_1 entre vecteurs d'intervalles et la deuxième méthode utilise la distance L_1 adaptative avec une seule composante en première variante, et avec deux composantes en deuxième variante. De Souza *et al.* (2004) ont proposé deux méthodes de nuées dynamiques en optimisant un critère basé sur une extension de la distance de Mahalanobis pour les données intervalles : dans la première méthode, la distance utilisée est adaptative et commune à toutes les classes, alors que dans la deuxième méthode chaque classe a sa propre distance adaptative. De Carvalho *et al.* (2006) ont appliqué l'algorithme des nuées dynamiques basé sur le carré de la distance L_2 pour les données intervalles et ont présenté trois techniques de normalisation pour les variables de type intervalle. Dans toutes ces méthodes basées sur l'algorithme des nuées dynamiques, les prototypes des classes sont des éléments de l'espace de représentation des objets à classer, c'est à dire des vecteurs dont les composantes sont des intervalles. Malgré la simplicité de leur implémentation et leur complexité algorithmique réduite, ces méthodes présentent quelques inconvénients : le nombre de classes doit être fourni *a priori*, les classes reconnues sont convexes (il est impossible de reconnaître des classes concentriques, par exemple), et la partition finale dépend du choix des prototypes initiaux.

Dans le cadre de la **classification floue**, De Carvalho (2007) a étendu l'algorithme des c-moyennes pour classifier des données intervalles en optimisant un critère basé sur le carré de la distance L_2.

Les **cartes auto-organisatrices de Kohonen *(SOM)*** sont aussi utilisées pour la classification de données intervalles tout en fournissant une information supplémentaire sur la proximité des classes (préservation de la topologie). Bock (2003) a construit une carte auto-organisatrice pour les données intervalles dont l'apprentissage se fait d'une façon séquentielle en utilisant la distance *vertex-type*. De Carvalho *et al.* ont adapté l'algorithme d'optimisation pour les cartes topologiques sur les données de type intervalle en utilisant des distances L_2 (De Carvalho et Pacifico, 2011) et L_1 (De Carvalho *et al.*, 2012) adaptatives et non adaptatives. Cabanes *et al.* (2013) ont étendu l'algorithme *S2L-SOM (Simultaneous Two-Levels-SOM)* aux données intervalles. Il s'agit d'un algorithme qui permet de faire une projection des données sur une grande carte tout en classifiant les vecteurs prototypes d'une façon simultanée. À la fin de l'exécution de l'algorithme, chaque observation appartiendra à la classe de son neurone vainqueur. L'avantage de cette méthode réside dans le fait que le nombre de classes est détecté automatiquement et des classes de formes quelconques sont reconnues. Il est important de noter que les résultats de classification obtenus avec les cartes auto-organisatrices sont sensibles aux paramètres d'apprentissage de la carte qui doivent être judicieusement choisis.

Vu que les données de type intervalle sont des données symboliques, il est possible d'appliquer les algorithmes de classification de données symboliques exposés dans le chapitre 2 pour classifier les données intervalles, comme à titre d'exemple, les cartes auto-organisatrices pour les dissimilarités entre les données (voir sous-section 2.4.3, p.65), en utilisant une mesure de proximité entre vecteurs d'intervalles (voir section 3.3, p.91). La classification hiérarchique ascendante avec une distance adéquate entre vecteurs d'intervalles peut aussi être utilisée.

Dans ce qui suit, nous allons exposer les algorithmes que nous proposons dans le cadre de cette thèse pour la classification de données intervalles avec les cartes auto-organisatrices.

FIGURE 3.2 – Méthodes de partitionnement existantes pour les données de type intervalle.

3.6 Cartes auto-organisatrices pour les données de type intervalle

Nous nous proposons de bâtir une carte auto-organisatrice dans le but de projeter l'ensemble des observations décrites par des variables de type intervalle. Nous supposons que les K neurones sont arrangés suivant une grille rectangulaire où l'emplacement $\boldsymbol{r}_k$ de chaque neurone k est dicté par le numéro de ligne et le numéro

de colonne du neurone sur la grille. De même, à chaque neurone k est associé un vecteur référent ou vecteur prototype $\boldsymbol{\mathcal{W}}_k$ de l'espace des observations défini par :

$$\boldsymbol{\mathcal{W}}_k = \big([u_k^1, v_k^1], \ldots, [u_k^p, v_k^p]\big)^T$$

Par la suite, l'ensemble des observations sera représenté par la matrice $\mathbb{R}$ de taille $n \times 2p$, et celui des vecteurs prototypes par la matrice $\mathbb{W}$ de taille $K \times 2p$. Dans ce qui suit, nous exposons l'algorithme d'apprentissage séquentiel d'une carte auto-organisatrice pour les données intervalles. Plus tard, nous proposons plusieurs algorithmes d'apprentissage en mode différé des cartes auto-organisatrices pour les données intervalles.

3.6.1 Apprentissage heuristique séquentiel

L'apprentissage se fait en présentant, à chaque itération, une observation choisie au hasard à la carte, et en déterminant ensuite son neurone vainqueur *(Best Matching Unit)*, le neurone dont le vecteur référent est le plus proche de l'observation présentée au sens d'une distance donnée. Bock (2003) a proposé d'utiliser la distance *vertex-type* pour la recherche du neurone vainqueur d'une observation décrite par des intervalles. Mais il est possible aussi d'utiliser d'autres distances comme la distance L_1 ou L_2 ou des distances basées sur la distance de Hausdorff. En désignant par $\boldsymbol{\mathcal{R}}(t) = \big([a^1(t), b^1(t)], \ldots, [a^p(t), b^p(t)]\big)^T$ l'observation présentée à l'itération t, la recherche de son neurone vainqueur $c(t)$ se fait de la façon suivante :

$$c(t) = \underset{k \in \{1,\ldots,K\}}{\arg\min}\ d\big(\boldsymbol{\mathcal{R}}(t), \boldsymbol{\mathcal{W}}_k(t)\big) \tag{3.9}$$

où d est une distance entre deux vecteurs d'intervalles.

La mise à jour des vecteurs référents se fait de façon à les rapprocher davantage de l'observation présentée en agissant séparément sur les bornes inférieures et sur les bornes supérieures des intervalles. Ce rapprochement est réalisé en respectant une notion de voisinage de façon à ce que le rapprochement soit maximal pour le neurone vainqueur, et décroisse au fur et à mesure de l'éloignement des autres neurones de ce neurone vainqueur. La mise à jour des vecteurs référents se fait comme suit :

$$\begin{aligned} u_k^j(t+1) &= u_k^j(t) + \alpha(t) h_{kc}(t)(a^j(t) - u_k^j(t)) \\ v_k^j(t+1) &= v_k^j(t) + \alpha(t) h_{kc}(t)(b^j(t) - v_k^j(t)) \\ &\text{pour } j \in \{1, \ldots, p\} \text{ et } k \in \{1, \ldots, K\} \end{aligned} \tag{3.10}$$

où :

- $h_{kc}(t)$ est la fonction de voisinage qui définit la quantité de variation à appliquer au vecteur référent du neurone k en fonction de son éloignement du neurone vainqueur $c(t)$. Elle dépend de l'emplacement des deux neurones et d'un rayon de voisinage. Une simple fonction de voisinage vaut 1 pour les neurones appartenant à la zone définie par le rayon de voisinage et vaut 0 dans les autres cas. Une fonction de voisinage plus flexible est la fonction de voisinage gaussienne (voir équation (1.19), p.27). Le rayon de voisinage, qui est l'écart-type de la fonction gaussienne, est suffisamment large au début de l'apprentissage pour inclure le plus grand nombre de neurones possible, et décroît avec les itérations pour n'inclure que le neurone vainqueur et ses voisins immédiats, ou même seulement le neurone vainqueur si le rayon de voisinage atteint des valeurs très faibles.

- $\alpha(t)$ est le pas de l'apprentissage qui décroît avec les itérations dans le but de ralentir au fur et à mesure la vitesse de l'apprentissage. Cette décroissance peut se faire linéairement ou exponentiellement.

L'algorithme 3.9 résume les étapes de l'apprentissage en mode séquentiel d'une carte auto-organisatrice pour des observations décrites par des variables intervalles.

Algorithme 3.9 Apprentissage séquentiel des cartes auto-organisatrices pour les données intervalles.

Entrées : Fournir :

- $\mathbb{R} = \{\square$ La matrice des observations$\}$
- lig, col $\{\square$ Dimensions de la carte (nombre de lignes, nombre de colonnes) $K = lig \cdot col\}$
- T $\{\square$ Nombre total d'itérations$\}$
- σ_{init}, σ_{final} $\{\square$ Valeurs initiale et finale du rayon de voisinage$\}$
- α_{init} $\{\square$ Valeur initiale du pas d'apprentissage$\}$

Sorties : Récupérer :

- $\mathbb{W}(T)$ $\{\square$ Vecteurs prototypes auto-organisés$\}$
- $P_T = \{C_1, \ldots, C_K\}$ $\{\square$ Partition finale$\}$

Initialisation :

$t \leftarrow 0$ $\{\square$ t : Itération courante$\}$

Initialiser les vecteurs prototypes $\mathbb{W}(0)$

Apprentissage :
Répéter
$\alpha(t) \leftarrow \alpha_{init}\Big(1 - \frac{t}{T}\Big)$
$\sigma(t) \leftarrow \sigma_{init} + \Big(\frac{t}{T}\Big)(\sigma_{final} - \sigma_{init})$
Choisir aléatoirement une observation $\boldsymbol{\mathcal{R}}(t) = \big([a^1(t), b^1(t)], \ldots, [a^p(t), b^p(t)]\big)^T$
Recherche de son neurone vainqueur (BMU) :
Pour $k = 1 \rightarrow K$ **Faire**
Calculer $d_k \leftarrow d\big(\boldsymbol{\mathcal{R}}(\boldsymbol{t}), \boldsymbol{\mathcal{W}}_k(t)\big)$
Fin pour
$c(t) \leftarrow \arg\min_{k \in \{1,\ldots,K\}} d_k$
Calcul des valeurs de la fonction de voisinage :
Pour $k = 1 \rightarrow K$ **Faire**
$h_{ck}(t) \leftarrow \exp\Big(-\frac{\|\boldsymbol{r}_c - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\Big)$
Fin pour
Mise à jour des vecteurs prototypes :
Pour $k = 1 \rightarrow K$ **Faire**
Pour $j = 1 \rightarrow p$ **Faire**
$u_k^j(t+1) = u_k^j(t) + \alpha(t) h_{kc}(t)(a^j(t) - u_k^j(t))$
$v_k^j(t+1) = v_k^j(t) + \alpha(t) h_{kc}(t)(b^j(t) - v_k^j(t))$
Fin pour
Fin pour
$t \leftarrow t + 1$
Jusqu'à $t > T$
Retourner $\mathbb{W}(T)$

3.6.2 Apprentissage en mode différé en optimisant un critère

Dans l'apprentissage en mode différé *(batch)* proposé par Kohonen, chaque itération de l'algorithme consiste à présenter toutes les observations à la carte, à déterminer le neurone vainqueur de chaque observation, et à mettre à jour les vecteurs prototypes des neurones de manière heuristique. Dans les algorithmes que nous allons proposer dans cette sous-section, l'apprentissage de la carte se fait en minimisant le

critère suivant :

$$G_{intSOM}(\mathbb{W}) = \sum_{i=1}^{n} \sum_{k=1}^{K} h_{kc_i} d(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k) \tag{3.11}$$

où h_{kc_i} est la fonction de voisinage pour les neurones k et le neurone vainqueur c_i de l'observation $\boldsymbol{\mathcal{R}}_i$, et d est une distance entre deux vecteurs d'intervalles.

Pour chaque itération t du processus d'apprentissage, un rayon de voisinage est fixé, les valeurs des fonctions de voisinage $h_{kc_i}(t)$ sont calculées et la détermination des vecteurs prototypes des neurones se fait en minimisant le critère G_{intSOM}. Donc, chaque itération du processus d'apprentissage peut engendrer plusieurs sous itérations jusqu'à ce que le critère G_{intSOM} ne change plus. Chaque itération ressemble ainsi à l'algorithme des nuées dynamiques (voir sous-section 1.3.2, p.24) avec la différence que les distances sont pondérées par les valeurs de la fonction de voisinage. Le rayon de voisinage est initialisé à une grande valeur (moitié de la plus grande dimension de la carte) pour activer le plus grand nombre de neurones possible, puis décroît avec les itérations pour atteindre une valeur faible à la fin de l'algorithme de façon à activer seulement le neurone vainqueur. Dans les expériences menées dans ce chapitre, nous proposons une décroissance linéaire du rayon de voisinage.

Heskes (1999b) a prouvé que pour minimiser le critère G_{intSOM}, la recherche du neurone vainqueur d'une observation $\boldsymbol{\mathcal{R}}_i$ doit se faire en calculant une somme de distances pondérées par les valeurs des fonctions de voisinage, tel que :

$$c_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \sum_{\tau=1}^{K} h_{k\tau} d\big(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_\tau\big) \tag{3.12}$$

plutôt qu'avec l'équation (3.9) (voir sous-section 1.3.3, p.35), où le neurone vainqueur d'une observation est celui dont le prototype se trouve à une distance minimale de cette observation.

La détermination d'un vecteur prototype $\boldsymbol{\mathcal{W}}_k$ se fait en minimisant le critère suivant :

$$g_k(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i} d(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k) \tag{3.13}$$

Ceci est réalisé en tenant compte de la distance ou de la dissimilarité choisie. Nous allons ci-après considérer plusieurs cas :

Carré de la distance L_2. Quand d est une dissimilarité définie suivant l'équation (3.2), le critère g_k devient :

$$g_{k(L_2^2)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i} \Big(\sum_{j=1}^{p} \big((a_i^j - u_k^j)^2 + (b_i^j - v_k^j)^2\big) \Big) \tag{3.14}$$

La minimisation de $g_{k(L_2^2)}$ est assurée en minimisant tous les $g^j_{k(L_2^2)}$, $j \in \{1, \ldots, p\}$ avec :

$$g^j_{k(L_2^2)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i} \big((a_i^j - u_k^j)^2\big) + \sum_{i=1}^{n} h_{kc_i} \big((b_i^j - v_k^j)^2\big) \tag{3.15}$$

Or, les u_k^j et v_k^j qui minimisent $g^j_{k(L_2^2)}$ sont la moyenne pondérée des a_i^j, $i \in \{1, \ldots, n\}$ et des b_i^j, $i \in \{1, \ldots, n\}$ respectivement, les poids étant les valeurs de la fonction de voisinage (voir annexe D, p.193).

$$\begin{aligned} u_k^j &= \frac{\sum_{i=1}^{n} h_{kc_i} a_i^j}{\sum_{i=1}^{n} h_{kc_i}} \\ v_k^j &= \frac{\sum_{i=1}^{n} h_{kc_i} b_i^j}{\sum_{i=1}^{n} h_{kc_i}} \end{aligned} \tag{3.16}$$

Distance L_1. Quand d est une distance définie suivant l'équation (3.3), le critère g_k devient (Hajjar et Hamdan, 2014) :

$$g_{k(L_1)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i} \Big(\sum_{j=1}^{p} \big(|a_i^j - u_k^j| + |b_i^j - v_k^j|\big) \Big) \tag{3.17}$$

La minimisation de $g_{k(L_1)}$ est assurée en minimisant tous les $g^j_{k(L_1)}$, $j \in \{1, \ldots, p\}$ avec :

$$g^j_{k(L_1)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i} \big(|a_i^j - u_k^j|\big) + \sum_{i=1}^{n} h_{kc_i} \big(|b_i^j - v_k^j|\big) \tag{3.18}$$

Or les u_k^j et v_k^j qui minimisent $g^j_{k(L_1)}$ sont la médiane pondérée des a_i^j, $i \in \{1, \ldots, n\}$ et des b_i^j, $i \in \{1, \ldots, n\}$ respectivement, les poids étant les valeurs de la fonction de voisinage (voir annexe C, p.191).

$$\begin{aligned} u_k^j &= \mathit{mediane\_ponderee}(a_i^j)\ i \in \{1, \ldots, n\} \\ v_k^j &= \mathit{mediane\_ponderee}(b_i^j)\ i \in \{1, \ldots, n\} \end{aligned} \tag{3.19}$$

Combinaison de type L_1 de la distance de Hausdorff. Quand d est une distance définie selon l'équation (3.4), le critère g_k devient :

$$g_{k\left(\text{Hausdorff-}L_1\right)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i}\Big(\sum_{j=1}^{p} \max\left(|a_i^j - u_k^j|, |b_i^j - v_k^j|\right)\Big) \tag{3.20}$$

La minimisation de $g_{k\left(\text{Hausdorff-}L_1\right)}$ est assurée en minimisant tous les $g^j_{k\left(\text{Hausdorff-}L_1\right)}$, $j \in \{1, \ldots, p\}$ avec :

$$g^j_{k\left(\text{Hausdorff-}L_1\right)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i}\Big(\max\left(|a_i^j - u_k^j|, |b_i^j - v_k^j|\right)\Big) \tag{3.21}$$

Soient $m_i^j = \frac{a_i^j + b_i^j}{2}$, $l_i^j = \frac{b_i^j - a_i^j}{2}$, $\mu_k^j = \frac{u_k^j + v_k^j}{2}$ et $\lambda_k^j = \frac{v_k^j - u_k^j}{2}$. En exprimant les bornes de chaque intervalle en fonction de son centre et de sa demi-longueur, $g^j_{k\left(\text{Hausdorff-}L_1\right)}$ devient :

$$\begin{aligned} g^j_{k\left(\text{Hausdorff-}L_1\right)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i}(t) max\Big(&\big|(\mu_k^j - \lambda_k^j) - (m_i^j - l_i^j)\big|, \\ &\big|(\mu_k^j + \lambda_k^j) - (m_i^j + l_i^j)\big|\Big) \\ g^j_{k\left(\text{Hausdorff-}L_1\right)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i}(t) max\Big(&\big|(\mu_k^j - m_i^j) - (\lambda_k^j - l_i^j)\big|, \\ &\big|(\mu_k^j - m_i^j) + (\lambda_k^j - l_i^j)\big|\Big) \end{aligned} \tag{3.22}$$

Or $max(\big|x - y\big|, \big|x + y\big|) = \big|x\big| + \big|y\big|$, donc $g^j_{k\left(\text{Hausdorff-}L_1\right)}$ est réduit à :

$$g^j_{k\left(\text{Hausdorff-}L_1\right)}(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i}(t)\Big|\mu_k^j - m_i^j\Big| + \sum_{i=1}^{n} h_{kc_i}(t)\Big|\lambda_k^j - l_i^j\Big| \tag{3.23}$$

μ_k^j et λ_k^j qui minimisent $g^j_{k(H)}$ valent la médiane pondérée des m_i^j, $i \in \{1, \ldots, n\}$) et des l_i^j, $i \in \{1, \ldots, n\}$ respectivement, les poids étant les valeurs de la fonction de voisinage (voir annexe C, p.191). Donc, la mise à jour des vecteurs prototypes se fait comme suit :

$$\begin{aligned} u_k^j &= mediane\_ponderee(m_i^j) - mediane\_ponderee(l_i^j) \;\; i \in \{1, \ldots, n\} \\ v_k^j &= mediane\_ponderee(m_i^j) + mediane\_ponderee(l_i^j) \;\; i \in \{1, \ldots, n\} \end{aligned} \tag{3.24}$$

L'algorithme 3.10 détaille les étapes du processus d'apprentissage d'une carte auto-organisatrice en mode différé, en optimisant un critère basé sur le carré de la distance L_2, sur la distance L_1 ou sur une combinaison de type L_1 de distances de Hausdorff. La recherche du neurone vainqueur constitue le mécanisme le plus coûteux de cet algorithme nécessitant un nombre d'opérations arithmétiques proportionnel à $S.n.K^2.p$ où S est le nombre total de sous-itérations. Donc, la complexité d'un tel algorithme est $O(S.n.K^2.p)$. C'est une complexité linéaire en fonction du nombre d'observations n mais quadratique en nombre de neurones K.

Algorithme 3.10 Apprentissage en mode différé des cartes auto-organisatrices pour les données intervalles en optimisant un critère (intSOM-DYN).

Entrées : Fournir :

- $\boldsymbol{\mathcal{R}}_i = \left([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p]\right)^T$, $i \in \{1, \ldots, n\}$ {□ Individus ou observations}
- lig, col {□ Dimensions de la carte, $K = lig \cdot col$}
- $\sigma_{init}, \sigma_{final}$ {□ Valeurs initiale et finale du rayon de voisinage}

Sorties : Récupérer :

- $\mathbb{W}(f)$ {□ Vecteurs prototypes auto-organisés}
- $P_f = \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante associée à la réduction du rayon de voisinage}

$s \leftarrow 0$ {□ s : Itération courante associée à la mise à jour des vecteurs prototypes}

Initialiser les vecteurs prototypes $\mathbb{W}(0)$

Initialiser le rayon de voisinage $\sigma(t)$ à σ_{init}

Choisir *distance* {□ La distance entre vecteurs d'intervalles : L_2 (carré de la distance L_2), L_1, Hausdorff-L_1}

Si *distance* == Hausdorff-L_1 **Alors**

Calculer $m_i^j = (a_i^j + b_i^j)/2$, $l_i^j = (b_i^j - a_i^j)/2$

Finsi

Apprentissage :

Répéter

Répéter

Calcul des valeurs de la fonction de voisinage : $h_{k\tau}(t)$, $k, \tau \in \{1, \ldots, K\}$

Pour $k = 1 \rightarrow K$ **Faire**

Pour $\tau = 1 \rightarrow K$ **Faire**

$h_{k\tau}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_k - \boldsymbol{r}_\tau\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Calcul des neurones vainqueurs des observations :

Si *distance* == L_2 **Alors**

Pour $i = 1 \rightarrow n$ **Faire**

$c_i \leftarrow \arg\min_{k\in\{1,\dots,K\}} \sum_{\tau=1}^{K} h_{k\tau}(t) \sum_{j=1}^{p} \left((a_i^j - u_\tau^j(s))^2 + (b_i^j - v_\tau^j(s))^2\right)$

$C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Attribution du vecteur $\boldsymbol{\mathcal{R}}_i$ à la classe C_{c_i}}

Fin pour

Sinon Si *distance* == L_1 **Alors**

Pour $i = 1 \rightarrow n$ **Faire**

$c_i \leftarrow \arg\min_{k\in\{1,\dots,K\}} \sum_{\tau=1}^{K} h_{k\tau}(t) \sum_{j=1}^{p} \left(\left|a_i^j - u_\tau^j(s)\right| + \left|b_i^j - v_\tau^j(s)\right|\right)$

$C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Attribution du vecteur $\boldsymbol{\mathcal{R}}_i$ à la classe C_{c_i}}

Fin pour

Sinon Si *distance* == Hausdorff-L_1 **Alors**

Pour $i = 1 \rightarrow n$ **Faire**

$c_i \leftarrow \arg\min_{k\in\{1,\dots,K\}} \sum_{\tau=1}^{K} h_{k\tau}(t) \sum_{j=1}^{p} \max\left(|a_i^j - u_\tau^j(s)|, |b_i^j - v_\tau^j(s)|\right)$

$C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Attribution du vecteur $\boldsymbol{\mathcal{R}}_i$ à la classe C_{c_i}}

Fin pour

Finsi

Partition P_s générée

Calcul des valeurs de la fonction de voisinage : h_{kc_i}, $i \in \{1,\dots,n\}$, $k \in \{1,\dots,K\}$

Pour $i = 1 \rightarrow n$ **Faire**

Pour $k = 1 \rightarrow K$ **Faire**

$h_{kc_i}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_i} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Mise à jour des vecteurs prototypes :

Si *distance* == L_2 **Alors**

Pour $k = 1 \rightarrow K$ **Faire**

Pour $j = 1 \rightarrow p$ **Faire**

$u_k^j(s+1) \leftarrow \frac{\sum_{i=1}^{n} h_{kc_i}(t) a_i^j}{\sum_{i=1}^{n} h_{kc_i}(t)}$

$v_k^j(s+1) \leftarrow \frac{\sum_{i=1}^{n} h_{kc_i}(t) b_i^j}{\sum_{i=1}^{n} h_{kc_i}(t)}$

Fin pour

Fin pour

Sinon Si *distance* == L_1 **Alors**

Pour $k = 1 \rightarrow K$ **Faire**
Pour $j = 1 \rightarrow p$ **Faire**
$u_k^j(s+1) = mediane\_ponderee(a_i^j)\ i \in \{1, \ldots, n\}$
$v_k^j(s+1) = mediane\_ponderee(b_i^j)\ i \in \{1, \ldots, n\}$
Fin pour
Fin pour
Sinon Si *distance* == Hausdorff-L_1 **Alors**
Pour $k = 1 \rightarrow K$ **Faire**
Pour $j = 1 \rightarrow p$ **Faire**
$u_k^j(s+1) = mediane\_ponderee(m_i^j) - mediane\_ponderee(l_i^j), i \in \{1, \ldots, n\}$
$v_k^j(s+1) = mediane\_ponderee(m_i^j) + mediane\_ponderee(l_i^j), i \in \{1, \ldots, n\}$
Fin pour
Fin pour
Finsi
$s \leftarrow s + 1$
Jusqu'à $\sum_{i=1}^{n} \sum_{k=1}^{K} h_{kc_i} d\big(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k(s)\big) == \sum_{i=1}^{n} \sum_{k=1}^{K} h_{kc_i} d\big(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k(s-1)\big)$
$t \leftarrow t + 1$
Réduire $\sigma(t)$
Jusqu'à $\sigma(t) < \sigma_{final}$
Retourner $\mathbb{W}(f) = \mathbb{W}(s)$
Retourner $P_f = P_s$

3.6.3 Apprentissage heuristique en mode différé

Dans cette série d'algorithmes, la mise à jour des vecteurs prototypes se fait d'une façon heuristique à la manière de l'algorithme de Kohonen. Chaque itération du processus d'apprentissage consiste à choisir un rayon de voisinage, à rechercher le neurone vainqueur de chaque observation, à calculer les valeurs des fonctions de voisinage et à calculer les vecteurs prototypes des neurones. La recherche du neurone vainqueur se fait suivant l'équation (3.9), et la mise à jour des vecteurs prototypes se fait suivant l'équation (3.16) pour la distance de type L_2 (Hajjar et Hamdan, 2011c; Hamdan et Hajjar, 2011; Hajjar et Hamdan, 2012a,b), l'équation (3.19) pour la distance de type L_1 (Hajjar et Hamdan, 2011a), ou l'équation (3.24) pour la combinaison de type L_1 de distances de Hausdorff (Hajjar et Hamdan, 2011b). L'apprentissage s'arrête quand le nombre total d'itérations T, fixé *a priori*, est atteint. La

quantité de réduction du rayon de voisinage à chaque itération est inversement proportionnelle au nombre total d'itérations. La détermination du rayon de voisinage à l'itération t se fait par :

$$\sigma(t) = \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$$

où σ_{init} et σ_{final} sont respectivement les valeurs initiales et finales du rayon de voisinage. L'algorithme 3.11, décrit les étapes d'apprentissage en mode différé d'une carte auto-organisatrice quand la recherche du neurone vainqueur se fait en utilisant le carré de la distance L_2, la distance L_1 ou la combinaison de type L_1 de distances de Hausdorff. La recherche du neurone vainqueur constitue le mécanisme le plus coûteux de ces algorithmes nécessitant un nombre d'opérations arithmétiques proportionnel à $T.n.K.p$. Donc, la complexité d'un tel algorithme est $O(T.n.K.p)$. C'est une complexité linéaire en fonction du nombre d'observations n et en fonction du nombre de neurones K.

Algorithme 3.11 Apprentissage heuristique en mode différé des cartes auto-organisatrices pour les données intervalles (intSOM).

Entrées : Fournir :

- $\boldsymbol{\mathcal{R}}_i = \left([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p]\right)^T$, $i \in \{1, \ldots, n\}$ {□ Individus ou observations}
- lig, col {□ Dimensions de la carte, $K = lig \cdot col$}
- $\sigma_{init}, \sigma_{final}$ {□ Valeurs initiale et finale du rayon de voisinage}
- T {□ Nombre total d'itérations}

Sorties : Récupérer :

- $\mathbb{W}(T)$ {□ Vecteurs prototypes auto-organisés}
- $P_T = \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Initialiser les vecteurs prototypes $\mathbb{W}(0)$

Choisir *distance* {□ La distance entre vecteurs d'intervalles : L_2 (carré de la distance L_2), L_1, Hausdorff-L_1}

Si *distance* == Hausdorff-L_1 **Alors**

Calculer $m_i^j = (a_i^j + b_i^j)/2$, $l_i^j = (b_i^j - a_i^j)/2$

Finsi

Apprentissage :

Répéter

$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$

Calcul des neurones vainqueurs des observations :
Si *distance* $==$ L_2 **Alors**
  Pour $i = 1 \rightarrow n$ **Faire**
    $c_i = \arg\min_{k\in\{1,\ldots,K\}} \sum_{j=1}^{p} \left((a_i^j - u_k^j)^2 + (b_i^j - v_k^j)^2\right)$
    $C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Affecter l'observation $\boldsymbol{\mathcal{R}}_i$ à la classe C_{c_i}}
  Fin pour
Sinon Si *distance* $==$ L_1 **Alors**
  Pour $i = 1 \rightarrow n$ **Faire**
    $c_i = \arg\min_{k\in\{1,\ldots,K\}} \sum_{j=1}^{p} \left(|a_i^j - u_k^j| + |b_i^j - v_k^j|\right)$
    $C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Affecter l'observation $\boldsymbol{\mathcal{R}}_i$ à la classe C_{c_i}}
  Fin pour
Sinon Si *distance* $==$ Hausdorff-L_1 **Alors**
  Pour $i = 1 \rightarrow n$ **Faire**
    $c_i = \arg\min_{k\in\{1,\ldots,K\}} \sum_{j=1}^{p} \max\left(|a_i^j - u_k^j| + |b_i^j - v_k^j|\right)$
    $C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Affecter l'observation $\boldsymbol{\mathcal{R}}_i$ à la classe C_{c_i}}
  Fin pour
Finsi
Partition P_t générée
Calcul des valeurs de la fonction de voisinage : h_{kc_i}, $i \in \{1,\ldots,n\}$, $k \in \{1,\ldots,K\}$
Pour $i = 1 \rightarrow n$ **Faire**
  Pour $k = 1 \rightarrow K$ **Faire**
    $h_{kc_i}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_i} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$
  Fin pour
Fin pour
Mise à jour des vecteurs prototypes :
Si *distance* $==$ L_2 **Alors**
  Pour $k = 1 \rightarrow K$ **Faire**
    Pour $j = 1 \rightarrow p$ **Faire**
      $u_k^j(t+1) \leftarrow \frac{\sum_{i=1}^{n} h_{kc_i}(t) a_i^j}{\sum_{i=1}^{n} h_{kc_i}(t)}$
      $v_k^j(t+1) \leftarrow \frac{\sum_{i=1}^{n} h_{kc_i}(t) b_i^j}{\sum_{i=1}^{n} h_{kc_i}(t)}$
    Fin pour
  Fin pour
Sinon Si *distance* $==$ L_1 **Alors**
  Pour $k = 1 \rightarrow K$ **Faire**

Pour $j = 1 \rightarrow p$ **Faire**
$u_k^j(t+1) \leftarrow mediane\_ponderee(a_i^j)\ i \in \{1, \ldots, n\}$
$v_k^j(t+1) \leftarrow mediane\_ponderee(b_i^j)\ i \in \{1, \ldots, n\}$
Fin pour
Fin pour
Sinon Si *distance* == Hausdorff-L_1 **Alors**
Pour $k = 1 \rightarrow K$ **Faire**
Pour $j = 1 \rightarrow p$ **Faire**
$u_k^j(t+1) = mediane\_ponderee(m_i^j) - mediane\_ponderee(l_i^j), i \in \{1, \ldots, n\}$
$v_k^j(t+1) = mediane\_ponderee(m_i^j) + mediane\_ponderee(l_i^j), i \in \{1, \ldots, n\}$
Fin pour
Fin pour
Finsi
$t \leftarrow t + 1$
Jusqu'à $t > T$
Retourner $\mathbb{W}(T)$
Retourner P_T

Dans ce qui suit, nous allons proposer deux algorithmes d'apprentissage heuristique en mode différé d'une carte auto-organisatrice pour les données intervalles en utilisant des distances entre vecteurs d'intervalles basées sur la distance de Mahalanobis.

Apprentissage heuristique en mode différé en se basant sur la distance de Mahalanobis

Les deux méthodes proposées utilisent des distances basées sur la distance de Mahalanobis pour la recherche du neurone vainqueur (Hajjar et Hamdan, 2013a,b). Dans la première méthode, une distance commune à toutes les classes est utilisée tout au long du processus d'apprentissage. Par contre, dans la deuxième méthode, le processus d'apprentissage utilise une distance commune à toutes les classes puis bascule vers une distance différente pour chaque classe dans ses dernières itérations. L'utilisation de la distance de Mahalanobis permet de prendre en compte la variabilité des variables et la corrélation entre les variables et favorise la recherche de classes de formes ellipsoïdales.

Recherche du neurone vainqueur. Soit c_i le neurone vainqueur de l'observation $\boldsymbol{\mathcal{R}}_i$. c_i est déterminé par :

$$\delta(\boldsymbol{\mathcal{R}}_i, \boldsymbol{W}_{c_i}) = \underbrace{\min}_{k=1,\ldots,K} \delta(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k) \tag{3.25}$$

où $\delta(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k)$ est une distance entre deux vecteurs d'intervalles calculée en utilisant la distance de Mahalanobis (De Souza *et al.*, 2004).

$$\delta(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k) = d_M^2(\boldsymbol{a}_i, \boldsymbol{u}_k) + d_M^2(\boldsymbol{b}_i, \boldsymbol{v}_k) \tag{3.26}$$

où

- $\boldsymbol{a}_i = (a_i^1, \ldots, a_i^p)^T$ est le vecteur des coordonnées du sommet inférieur de $\boldsymbol{\mathcal{R}}_i$.
- $\boldsymbol{b}_i = (b_i^1, \ldots, b_i^p)^T$ est le vecteur des coordonnées du sommet supérieur de $\boldsymbol{\mathcal{R}}_i$.
- $\boldsymbol{u}_k = (u_k^1, \ldots, u_k^p)^T$ est le vecteur des coordonnées du sommet inférieur de $\boldsymbol{\mathcal{W}}_k$.
- $\boldsymbol{v}_k = (v_k^1, \ldots, v_k^p)^T$ est le vecteur des coordonnées du sommet supérieur de $\boldsymbol{W}_k$
- d_M^2 est le carré de la distance de Mahalanobis définie par :

$$\begin{aligned} d_M^2(\boldsymbol{a}_i, \boldsymbol{u}_k) &= (\boldsymbol{a}_i - \boldsymbol{u}_k)^T \boldsymbol{M}_L (\boldsymbol{a}_i - \boldsymbol{u}_k) \\ d_M^2(\boldsymbol{b}_i, \boldsymbol{v}_k) &= (\boldsymbol{b}_i - \boldsymbol{v}_k)^T \boldsymbol{M}_U (\boldsymbol{b}_i - \boldsymbol{v}_k) \end{aligned} \tag{3.27}$$

Les matrices $\boldsymbol{M}_L$ et $\boldsymbol{M}_U$ sont définies comme suit :

$$\begin{aligned} \boldsymbol{M}_L &= \big(det(\boldsymbol{Q}_{poolL})\big)^{1/p} \boldsymbol{Q}_{poolL}^{-1} \ , \ det(\boldsymbol{Q}_{poolL}) \neq 0 \\ \boldsymbol{M}_U &= \big(det(\boldsymbol{Q}_{poolU})\big)^{1/p} \boldsymbol{Q}_{poolU}^{-1} \ , \ det(\boldsymbol{Q}_{poolU}) \neq 0 \end{aligned} \tag{3.28}$$

où $\boldsymbol{Q}_{poolL}$ et $\boldsymbol{Q}_{poolU}$ sont les matrices de covariances communes entre toutes les classes :

$$\begin{aligned} \boldsymbol{Q}_{poolL} &= \frac{(n_1 - 1)\boldsymbol{S}_{1L} + \ldots + (n_K - 1)\boldsymbol{S}_{KL}}{n_1 + \ldots + n_K - K} \\ \boldsymbol{Q}_{poolU} &= \frac{(n_1 - 1)\boldsymbol{S}_{1U} + \ldots + (n_K - 1)\boldsymbol{S}_{KU}}{n_1 + \ldots + n_K - K} \end{aligned} \tag{3.29}$$

où $\boldsymbol{S}_{kL}$ est la matrice de covariance de l'ensemble des vecteurs $\{\boldsymbol{a}_i / i \in C_k\}$, et $\boldsymbol{S}_{kU}$ est la matrice de covariance de l'ensemble des vecteurs $\{\boldsymbol{b}_i / i \in C_k\}$. n_k est le cardinal de C_k.

Du fait que les matrices $\boldsymbol{M}_L$ et $\boldsymbol{M}_U$ sont les mêmes pour toutes les classes, la distance utilisée est commune à toutes les classes.

Une autre distance consiste à remplacer les matrices $\boldsymbol{M}_L$ et $\boldsymbol{M}_U$ par des matrices qui changent pour chaque classe, conduisant alors à une distance différente par classe :

$$\delta(\boldsymbol{\mathcal{R}}_i, \boldsymbol{W}_{c_i}) = \underbrace{min}_{k=1,\ldots,K} \big((\boldsymbol{a}_i - \boldsymbol{u}_k)^T \boldsymbol{M}_{kL}(\boldsymbol{a}_i - \boldsymbol{u}_k) + (\boldsymbol{b}_i - \boldsymbol{v}_k)^T \boldsymbol{M}_{kU}(\boldsymbol{b}_i - \boldsymbol{v}_k)\big) \tag{3.30}$$

Les matrices $\boldsymbol{M}_{kL}$ et $\boldsymbol{M}_{kU}$ associées àla classe C_k sont exprimées comme suit :

$$\begin{aligned} \boldsymbol{M}_{kL} &= \big(det(\boldsymbol{Q}_{kL})\big)^{1/p} \boldsymbol{Q}_{kL}^{-1} \ , \ det(\boldsymbol{Q}_{kL}) \neq 0 \\ \boldsymbol{M}_{kU} &= \big(det(\boldsymbol{Q}_{kU})\big)^{1/p} \boldsymbol{Q}_{kU}^{-1} \ , \ det(\boldsymbol{Q}_{kU}) \neq 0 \end{aligned} \tag{3.31}$$

où $\boldsymbol{Q}_{kL}$ est la matrice de covariance de l'ensemble des vecteurs $\{\boldsymbol{a}_i / i \in C_k\}$, et $\boldsymbol{Q}_{kU}$ la matrice de covariance de l'ensemble des vecteurs $\{\boldsymbol{b}_i / i \in C_k\}$.

Mise à jour des vecteurs prototypes. À chaque itération t, toutes les observations sont présentées à la carte, et le neurone vainqueur de chaque observation est déterminé. Ensuite chaque vecteur prototype $\boldsymbol{\mathcal{W}}_k$ est remplacé par la moyenne pondérée des vecteurs d'entrée où les poids sont les valeurs de la fonction de voisinage. L'équation de mise à jour des vecteurs prototypes est donnée par :

$$\boldsymbol{\mathcal{W}}_k(t+1) = \frac{\sum_{i=1}^{n} h_{kc_i}\big(\sigma(t)\big)\boldsymbol{\mathcal{R}}_i}{\sum_{i=1}^{n} h_{kc_i}\big(\sigma(t)\big)} \quad k \in \{1, \ldots, K\} \tag{3.32}$$

où $h_{kc_i}\big(\sigma(t)\big)$ est la fonction de voisinage gaussienne du neurone k et du neurone vainqueur c_i de l'observation $\boldsymbol{\mathcal{R}}_i$.

Dans les dernières itérations du processus d'apprentissage, quand le rayon de voisinage tend vers 0, les observations seront classifiées en K classes. Le prototype de chaque classe est le neurone k dont le vecteur prototype est $\boldsymbol{\mathcal{W}}_k$ est la moyenne des observations ayant le neurone k comme neurone vainqueur. Donc, les deux algorithmes dans leurs dernières itérations se réduisent à celui des nuées dynamiques. D'après (Govaert, 1975), le fait que le déterminant des matrices $\boldsymbol{M}_L$, $\boldsymbol{M}_U$, $\boldsymbol{M}_{kL}$

et $\boldsymbol{M}_{kU}$ vaut 1, et le fait que la mise à jour des vecteurs prototypes se fait suivant l'équation (3.32), le critère des nuées dynamiques G_{dyn}, défini dans l'équation (3.33), sera minimisé à la convergence des algorithmes.

$$G_{dyn} = \sum_{k=1}^{K} \sum_{\boldsymbol{\mathcal{R}}_i \in C_k} \delta(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k) \tag{3.33}$$

Algorithmes d'apprentissage Dans ce qui suit, nous présentons deux algorithmes d'apprentissage des cartes auto-organisatrices pour les données de type intervalle. Dans les deux algorithmes, la recherche du neurone vainqueur se fait en utilisant une distance entre vecteurs d'intervalles basée sur la distance de Mahalanobis. Le premier algorithme (intSOM-CMahalanobis) utilise une distance commune pour toutes les classes (voir équation (3.27), p.112) durant tout le processus d'apprentissage. Le deuxième algorithme (intSOM-DMahalanobis) est divisé en deux phases. Dans la première phase, une distance commune est utilisée pour toutes les classes (voir équation (3.27), p.112), alors que dans la deuxième phase une distance différente par classe est utilisée (voir équation (3.30), p.113). Le nombre d'itérations de la première phase constitue 90% du nombre total d'itérations. L'utilisation de ces deux phases va permettre une meilleure classification qu'avec la première méthode, tout en contournant les problèmes qui risquent de survenir au cas où une distance différente par classe est utilisée dès le début de l'apprentissage. Les algorithmes 3.12 et 3.13 présentent les étapes des méthodes intSOM-CMahalanobis et intSOM-DMahalanobis respectivement.

Algorithme 3.12 Apprentissage en mode différé des cartes auto-organisatrices pour les données intervalles avec une distance commune pour toutes les classes basée sur la distance de Mahalanobis (intSOM-CMahalanobis).

Entrées : Fournir :

- $\boldsymbol{\mathcal{R}}_i = \left([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p]\right)^T$, $i \in \{1, \ldots, n\}$ {□ Individus ou observations}
- lig, col {□ Dimensions de la carte, $K = lig \cdot col$}
- $\sigma_{init}, \sigma_{final}$ {□ Valeurs initiale et finale du rayon de voisinage}
- T {□ Nombre total d'itérations}

Sorties : Récupérer :

- $\mathbb{W}(T)$ {□ Vecteurs prototypes auto-organisés}
- $P_T = \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Générer aléatoirement une partition initiale $P_0 \leftarrow \{C_1, \ldots, C_K\}$

Calculer les vecteurs prototypes. Chaque prototype $\boldsymbol{\mathcal{W}}_k(0)$ est la moyenne des observations de sa classe C_k.

Initialiser le rayon de voisinage $\sigma(t)$ à σ_{init}

Déterminer $\boldsymbol{a}_i \leftarrow (a_i^1, \ldots, a_i^p)^T$ et $\boldsymbol{b}_i \leftarrow (b_i^1, \ldots, b_i^p)^T$

Apprentissage :

Répéter

$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$

Pour $k = 1 \rightarrow K$ **Faire**

Calculer $\boldsymbol{S}_{kL}$ {□ Matrice de covariance des $\boldsymbol{a}_i$ tel que $\boldsymbol{\mathcal{R}}_i \in C_k$}

Calculer $\boldsymbol{S}_{kU}$ {□ Matrice de covariance des $\boldsymbol{b}_i$ tel que $\boldsymbol{\mathcal{R}}_i \in C_k$}

Fin pour

Calcul des matrices de covariance communes entre toutes les classes :

$\boldsymbol{Q}_{poolL} \leftarrow \frac{(n_1-1)\boldsymbol{S}_{1L}+\ldots+(n_K-1)\boldsymbol{S}_{KL}}{n_1+\ldots+n_K-K}$

$\boldsymbol{Q}_{poolU} \leftarrow \frac{(n_1-1)\boldsymbol{S}_{1U}+\ldots+(n_K-1)\boldsymbol{S}_{KU}}{n_1+\ldots+n_K-K}$

$\boldsymbol{M}_L \leftarrow \big(det(\boldsymbol{Q}_{poolL})\big)^{1/p}\boldsymbol{Q}_{poolL}^{-1}$, $det(\boldsymbol{Q}_{poolL}) \neq 0$

$\boldsymbol{M}_U \leftarrow \big(det(\boldsymbol{Q}_{poolU})\big)^{1/p}\boldsymbol{Q}_{poolU}^{-1}$, $det(\boldsymbol{Q}_{poolU}) \neq 0$

Calcul des neurones vainqueurs des observations :

Pour $i = 1 \rightarrow n$ **Faire**

$c_i \leftarrow \arg\min_{k\in\{1,\ldots,K\}} \Big(\big(\boldsymbol{a}_i - \boldsymbol{u}_k\big)^T \boldsymbol{M}_L\big(\boldsymbol{a}_i - \boldsymbol{u}_k\big) + \big(\boldsymbol{b}_i - \boldsymbol{v}_k\big)^T \boldsymbol{M}_U\big(\boldsymbol{b}_i - \boldsymbol{v}_k\big)\Big)$

$C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Affecter l'observation $\boldsymbol{\mathcal{R}}_i$ au neurone c_i}

Fin pour

Partition P_t générée

Calcul des valeurs de la fonction de voisinage : h_{kc_i}, $i \in \{1, \ldots, n\}$, $k \in \{1, \ldots, K\}$

Pour $i = 1 \rightarrow n$ **Faire**

Pour $k = 1 \rightarrow K$ **Faire**

$h_{kc_i}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_i} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Mise à jour des vecteurs prototypes :

Pour $k = 1 \rightarrow K$ **Faire**

Pour $j = 1 \rightarrow p$ **Faire**

$u_k^j(t+1) \leftarrow \frac{\sum_{i=1}^n h_{kc_i}(t) a_i^j}{\sum_{i=1}^n h_{kc_i}(t)}$

$v_k^j(t+1) \leftarrow \frac{\sum_{i=1}^n h_{kc_i}(t) b_i^j}{\sum_{i=1}^n h_{kc_i}(t)}$

Fin pour

Fin pour

$t \leftarrow t+1$

Jusqu'à $t > T$

Retourner $\mathbb{W}(T)$

Retourner P_T

Algorithme 3.13 Apprentissage en mode différé des cartes auto-organisatrices pour les données intervalles avec une distance différente par classe basée sur la distance de Mahalanobis (intSOM-DMahalanobis).

Entrées : Fournir :

- $\boldsymbol{\mathcal{R}}_i = \left([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p]\right)^T$, $i \in \{1, \ldots, n\}$ {□ Individus ou observations}
- lig, col {□ Dimensions de la carte, $K = lig \cdot col$}
- $\sigma_{init}, \sigma_{final}$ {□ Valeurs initiale et finale du rayon de voisinage}
- T {□ Nombre total d'itérations}
- T_1 {Nombre d'itérations de la première phase}

Sorties : Récupérer :

- $\mathbb{W}(T)$ {□ Vecteurs prototypes auto-organisés}
- $P_T = \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Générer aléatoirement une partition initiale $P_0 \leftarrow \{C_1, \ldots, C_K\}$

Calculer les vecteurs prototypes. Chaque prototype $\boldsymbol{\mathcal{W}}_k(0)$ est la moyenne des observations de sa classe C_k.

Initialiser le rayon de voisinage $\sigma(t)$ à σ_{init}

Déterminer $\boldsymbol{a}_i \leftarrow (a_i^1, \ldots, a_i^p)^T$ et $\boldsymbol{b}_i \leftarrow (b_i^1, \ldots, b_i^p)^T$

Apprentissage :

Répéter

$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$

Pour $k = 1 \rightarrow K$ **Faire**

Calculer $\boldsymbol{S}_{kL}$ {□ Matrice de covariance des $\boldsymbol{a}_i$ tel que $\boldsymbol{\mathcal{R}}_i \in C_k$}

Calculer $\boldsymbol{S}_{kU}$ {□ Matrice de covariance des $\boldsymbol{b}_i$ tel que $\boldsymbol{\mathcal{R}}_i \in C_k$}

Fin pour

Si $t < T_1$ **Alors**

Calcul des matrices de covariance communes entre toutes les classes :

$\boldsymbol{Q}_{poolL} \leftarrow \frac{(n_1-1)\boldsymbol{S}_{1L}+\ldots+(n_K-1)\boldsymbol{S}_{KL}}{n_1+\ldots+n_K-K}$

$\boldsymbol{Q}_{poolU} \leftarrow \frac{(n_1-1)\boldsymbol{S}_{1U}+\ldots+(n_K-1)\boldsymbol{S}_{KU}}{n_1+\ldots+n_K-K}$

$\boldsymbol{M}_L \leftarrow \big(det(\boldsymbol{Q}_{poolL})\big)^{1/p}\boldsymbol{Q}_{poolL}^{-1}\ ,\ det(\boldsymbol{Q}_{poolL}) \neq 0$

$\boldsymbol{M}_U \leftarrow \big(det(\boldsymbol{Q}_{poolU})\big)^{1/p}\boldsymbol{Q}_{poolU}^{-1}\ ,\ det(\boldsymbol{Q}_{poolU}) \neq 0$

Calcul des neurones vainqueurs des observations :

Pour $i = 1 \rightarrow n$ **Faire**

$c_i \leftarrow \arg\min_{k\in\{1,\ldots,K\}} \Big(\big(\boldsymbol{a}_i - \boldsymbol{u}_k\big)^T \boldsymbol{M}_L \big(\boldsymbol{a}_i - \boldsymbol{u}_k\big) + \big(\boldsymbol{b}_i - \boldsymbol{v}_k\big)^T \boldsymbol{M}_U \big(\boldsymbol{b}_i - \boldsymbol{v}_k\big) \Big)$

$C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Affecter l'observation $\boldsymbol{\mathcal{R}}_i$ à la classe C_{c_i}}

Fin pour

Sinon

$\boldsymbol{Q}_{kL} \leftarrow \boldsymbol{S}_{kL}$

$\boldsymbol{Q}_{kU} \leftarrow \boldsymbol{S}_{kU}$

$\boldsymbol{M}_{kL} \leftarrow \big(det(\boldsymbol{Q}_{kL})\big)^{1/p}\boldsymbol{Q}_{kL}^{-1}\ ,\ det(\boldsymbol{Q}_{kL}) \neq 0$

$\boldsymbol{M}_{kU} \leftarrow \big(det(\boldsymbol{Q}_{kU})\big)^{1/p}\boldsymbol{Q}_{kU}^{-1}\ ,\ det(\boldsymbol{Q}_{kU}) \neq 0$

Calcul des neurones vainqueurs des observations :

Pour $i = 1 \rightarrow n$ **Faire**

$c_i \leftarrow \arg\min_{k\in\{1,\ldots,K\}} \Big(\big(\boldsymbol{a}_i - \boldsymbol{u}_k\big)^T \boldsymbol{M}_{kL} \big(\boldsymbol{a}_i - \boldsymbol{u}_k\big) + \big(\boldsymbol{b}_i - \boldsymbol{v}_k\big)^T \boldsymbol{M}_{kU} \big(\boldsymbol{b}_i - \boldsymbol{v}_k\big) \Big)$

$C_{c_i} \leftarrow \boldsymbol{\mathcal{R}}_i$ {□ Affecter l'observation $\boldsymbol{\mathcal{R}}_i$ au neurone c_i}

Fin pour

Finsi

Partition P_t générée

Calcul des valeurs de la fonction de voisinage : h_{kc_i}, $i \in \{1,\ldots,n\}$, $k \in \{1,\ldots,K\}$

Pour $i = 1 \rightarrow n$ **Faire**

Pour $k = 1 \rightarrow K$ **Faire**

$h_{kc_i}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_i} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Mise à jour des vecteurs prototypes :

Pour $k = 1 \rightarrow K$ **Faire**

Pour $j = 1 \rightarrow p$ **Faire**

$u_k^j(t+1) \leftarrow \frac{\sum_{i=1}^n h_{kc_i}(t) a_i^j}{\sum_{i=1}^n h_{kc_i}(t)}$

$v_k^j(t+1) \leftarrow \frac{\sum_{i=1}^n h_{kc_i}(t) b_i^j}{\sum_{i=1}^n h_{kc_i}(t)}$

Fin pour
Fin pour
$t \leftarrow t + 1$
Jusqu'à $t > T$
Retourner $\mathbb{W}(T)$
Retourner P_T

3.7 Étude expérimentale

Pour tester les méthodes proposées dans ce chapitre, nous avons utilisé un jeu de données intervalles simulées et trois jeux de données intervalles réelles, consistant en deux jeux de données réelles que nous avons construits dans le cadre de cette thèse, et un jeu de données intervalles regroupant des informations concernant des modèles de voitures (De Carvalho *et al.*, 2006). Le premier jeu de données construit concerne les températures minimales et maximales enregistrées dans les stations météorologiques françaises tout au long de l'année 2010. Le deuxième jeu de données construit concerne les températures minimales et maximales enregistrées dans les stations météorologiques libanaises tout au long de l'année 2010.

Pour évaluer la performance des cartes auto-organisatrices issues des expériences menées, nous évaluons les capacités de classification et les capacités de préservation de la topologie.
Dans le cas où une partition *a priori* existe pour un jeu de données, les résultats de la classification pourront être évalués en utilisant l'un des critères externes suivant : l'indice de Rand corrigé *(CR Index)* et/ou le taux global de l'erreur de classification *(OERC)* (voir sous-section 1.4.1, p.39).
La préservation de la topologie est évaluée en calculant l'erreur topographique (te) (voir équation (1.24), p.34).
Il est possible aussi de visualiser les vecteurs prototypes (en les connectant par leurs centres), ainsi que les vecteurs de données, sur un même graphique dans le but d'évaluer visuellement le déploiement de la carte sur les données et la préservation de la topologie (chaque vecteur prototype doit être connecté à ses voisins). Dans le cas où la dimension des données est supérieure à 2, une analyse en composantes principales pour les données intervalles est requise pour projeter les vecteurs de données et les vecteurs prototypes sur un espace bidimensionnel (voir annexe B, p.189).

3.7.1 Jeu de données simulées

Le jeu de données simulées consiste en 1800 observations décrites par deux variables de type intervalle générées à l'aide de 9 gaussiennes bivariées de formes elliptiques. Ayant une partition *a priori*, ce jeu de données est utilisé pour tester les capacités de classification d'une carte auto-organisatrice quand l'apprentissage se fait suivant les méthodes intSOM-DMahalanobis, intSOM-CMahalanobis ainsi qu'avec la méthode intSOM (voir algorithme 3.11, p.109) associée au carré de la distance L_2 que nous notons par la suite intSOM-L_2 .

Paramètres de simulation. Afin de simuler un jeu de données intervalles, nous commençons par simuler un jeu de données de n points $\boldsymbol{x}_i \in I\!R^2$, ensuite, nous rendons leurs positions imprécises en générant $\tilde{\boldsymbol{x}}_i$. Finalement, nous construisons les rectangles représentant les intervalles ayant comme centres $\tilde{\boldsymbol{x}}_i$ (Hamdan, 2005).

Le jeu de données artificielles comprend 1800 points également distribués en 9 classes (voir figure 3.3, p.120). Chaque classe C_k est générée à l'aide d'une distribution normale bivariée de moyenne $\boldsymbol{\mu}_k = (\mu_k^1, \mu_k^2)$ et de matrice de covariance $\boldsymbol{\Sigma}_k$:

$$\boldsymbol{\Sigma}_k = \begin{pmatrix} {\delta_k^1}^2 & \rho_k \delta_k^1 \delta_k^2 \\ \rho_k \delta_k^1 \delta_k^2 & {\delta_k^2}^2 \end{pmatrix}$$

La moyenne et la matrice de covariance de chaque classe simulée sont :
$C_1 : \mu_1^1 = 4,\ \mu_1^2 = 13,\ {\delta_1^1}^2 = 9,\ {\delta_1^2}^2 = 1/9,\ \rho_1 = 0.$
$C_2 : \mu_2^1 = 4,\ \mu_2^2 = 8.5,\ {\delta_2^1}^2 = 2.33,\ {\delta_2^2}^2 = 6.78,\ \rho_2 = 0.97.$
$C_3 : \mu_3^1 = 4,\ \mu_3^2 = 5,\ {\delta_3^1}^2 = 9,\ {\delta_3^2}^2 = 1/9,\ \rho_3 = 0.$
$C_4 : \mu_4^1 = 15,\ \mu_4^2 = 13,\ {\delta_4^1}^2 = 9,\ {\delta_4^2}^2 = 1/9,\ \rho_4 = 0.$
$C_5 : \mu_5^1 = 15,\ \mu_5^2 = 8.5,\ {\delta_5^1}^2 = 2.33,\ {\delta_5^2}^2 = 6.78,\ \rho_5 = 0.97.$
$C_6 : \mu_6^1 = 15,\ \mu_6^2 = 5,\ {\delta_6^1}^2 = 9,\ {\delta_6^2}^2 = 1/9,\ \rho_6 = 0.$
$C_7 : \mu_7^1 = 26,\ \mu_7^2 = 13,\ {\delta_7^1}^2 = 9,\ {\delta_7^2}^2 = 1/9,\ \rho_7 = 0.$
$C_8 : \mu_8^1 = 26,\ \mu_8^2 = 8.5,\ {\delta_8^1}^2 = 2.33,\ {\delta_8^2}^2 = 6.78,\ \rho_8 = 0.97.$
$C_9 : \mu_9^1 = 26,\ \mu_9^2 = 5,\ {\delta_9^1}^2 = 9,\ {\delta_9^2}^2 = 1/9,\ \rho_9 = 0.$

Les classes obtenues de cette simulation sont mélangées à 7.6%.

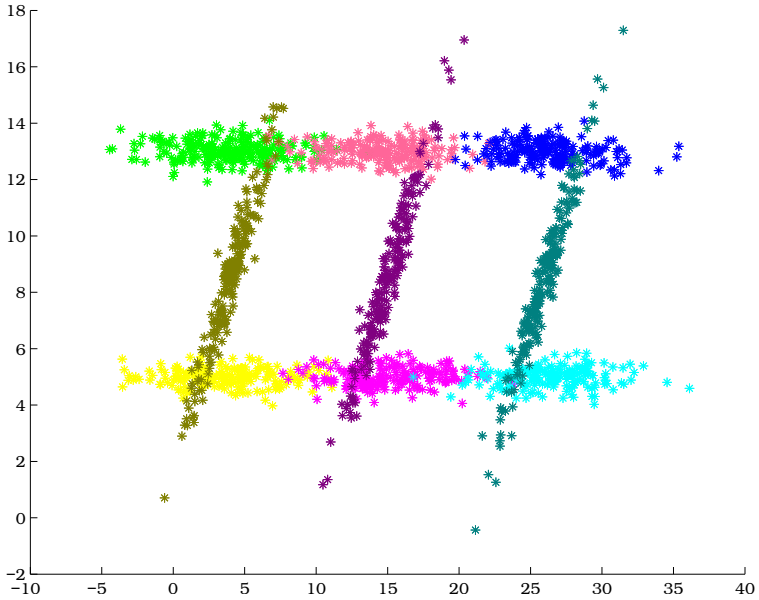


FIGURE 3.3 – Jeu de données simulées ponctuelles.

Les données imprécises $\tilde{\boldsymbol{x}}_i$ (variable aléatoire $\tilde{X}_i$) sont calculées en ajoutant des perturbations $\boldsymbol{e}_i$ (variable aléatoire E_i) à $\boldsymbol{x}_i$ (variable aléatoire X_i) : $\tilde{X}_i = X_i + E_i$ où $E_i \sim \mathcal{N}(0, diag(\varphi_{1i}^2, \varphi_{2i}^2))$, et φ_{1i}^2, φ_{2i}^2 suivent une loi gamma inverse $\mathcal{IG}(a, 1)$ (en s'inspirant du choix de la distribution *a priori* sur les paramètres d'une distribution gaussienne dans le cadre de l'inférence bayésienne (Hamdan, 2005)).

Ayant $\tilde{\boldsymbol{x}}_i = (\tilde{x}_i^1, \tilde{x}_i^2)^T$, φ_{1i}^2 et φ_{2i}^2, les intervalles $\boldsymbol{\mathcal{R}}_i$ sont construits comme suit :

$$\boldsymbol{\mathcal{R}}_i = \left([\tilde{x}_i^1 - 2\varphi_{1i}, \tilde{x}_i^1 + 2\varphi_{1i}], [\tilde{x}_i^2 - 2\varphi_{2i}, \tilde{x}_i^2 + 2\varphi_{2i}]\right)^T \quad i \in \{1, \ldots, n\} \tag{3.34}$$

Le paramètre a de la loi gamma inverse contrôle la longueur des intervalles simulés. Nous choisissions pour a les valeurs 7.11, 2.5 et 1.7, qui correspondent respectivement aux longueurs moyennes de 1.5, 3 et 4 pour les intervalles. Donc, pour chaque valeur de a, un jeu de données intervalles est généré.

Paramètres de l'apprentissage. Nous proposons de bâtir, pour ce jeu de données, une carte auto-organisatrice de $K = 9$ neurones arrangés suivant une grille carrée. L'apprentissage de la carte se fait suivant trois méthodes : la méthode intSOM-

DMahalanobis (voir algorithme 3.13, p.116), la méthode intSOM-CMahalanobis (voir algorithme 3.12, p.114) et la méthode intSOM-L_2 (voir algorithme 3.11, p.109).

Le rayon de voisinage initial est initialisé à $\sigma_{init} = 3$ et décroît jusqu'à $\sigma_{final} = 0.1$. La partition finale dépend des vecteurs prototypes initiaux. Afin de garantir un choix adéquat des prototypes initiaux, 20 lancées de chaque algorithme sont exécutées, chacune correspondant à un ensemble différent de prototypes initiaux, et les résultats produits par la lancée qui permet de minimiser le plus le critère G_{DYN} (voir équation (3.33), p.114) sont adoptés. Le nombre total d'itérations est $T = 600$, alors que le nombre d'itérations de la première phase de l'algorithme intSOM-DMahalanobis vaut $T1 = 540$, ce qui correspond à 90% du nombre total d'itérations.

Résultats. Les figures 3.4, 3.5 et 3.6 montrent les résultats obtenus après apprentissage de la carte sur un jeu de données intervalles avec une longueur d'intervalle moyenne de $L = 1.5$ ($a = 7.11$), en utilisant respectivement les méthodes intSOM-DMahalanobis, intSOM-CMahalanobis et intSOM-L_2. Les observations appartenant à la même classe sont dessinées avec une même couleur. Les vecteurs prototypes, connectés par leurs centres, sont dessinés en rouge. Nous remarquons que la partition obtenue avec la méthode intSOM-DMahalanobis correspond au mieux à la partition *a priori* de ce jeu de données, du fait que la distance utilisée dans les dernières itérations du processus d'apprentissage est une distance adaptative et différente pour chaque classe. Nous remarquons aussi que la méthode intSOM-CMahalanobis classifie les données plus correctement que la méthode intSOM-L_2. D'ailleurs, ce résultat est prévu du fait que la distance L_2 contribue à la reconnaissance de classes sphériques en ignorant la corrélation entre les variables et en supposant que toutes les variables sont de même importance. À propos de l'organisation des vecteurs prototypes, nous remarquons que chacun d'entre eux est relié aux vecteurs qui lui sont voisins, ce qui prouve que la topologie est parfaitement préservée.

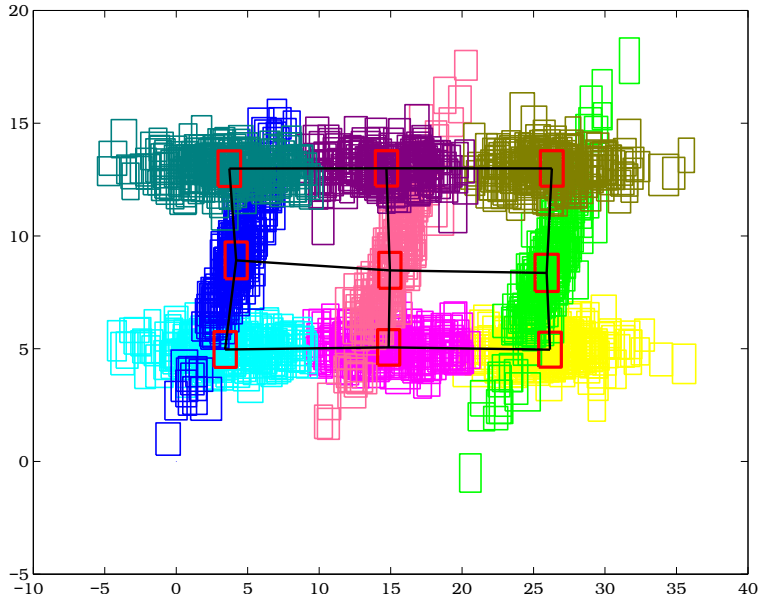


FIGURE 3.4 – Données et carte après apprentissage pour le jeu de données simulées avec la méthode intSOM-DMahalanobis.

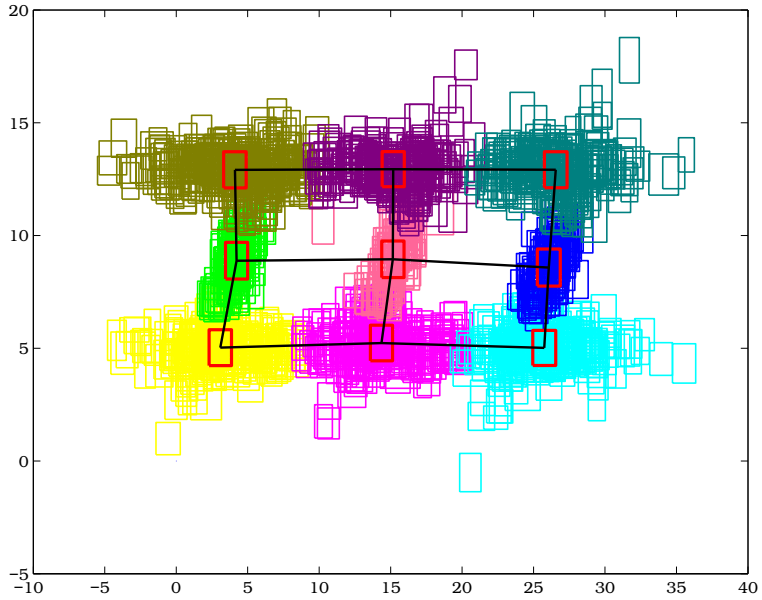


FIGURE 3.5 – Données et carte après apprentissage pour le jeu de données simulées avec la méthode intSOM-CMahalanobis.

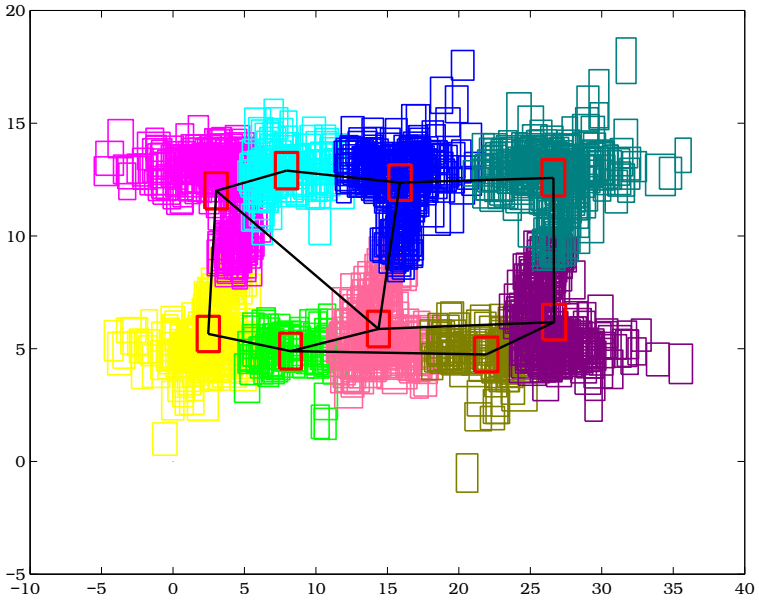


FIGURE 3.6 – Données et carte après apprentissage pour le jeu de données simulées avec la méthode intSOM-L_2.

Dans le but de fournir des résultats de classification plus concluantes, nous avons simulé 20 réplications du jeu de données ponctuelles, et pour chaque réplication, nous avons généré trois jeux de données intervalles chacun correspondant à une longueur moyenne différente pour les intervalles. Pour chaque jeu de données intervalles, nous avons évalué la qualité de la carte en calculant l'erreur topographique (te) ainsi que le taux global d'erreur de classification ($OERC$). Le tableau 3.2 donne la moyenne et l'écart-type (entre parenthèses) du te et du $OERC$ sur les 20 réplications (simulations de Monte Carlo), par méthode et par longueur moyenne d'intervalle.

Tableau 3.2 – Évaluation de la qualité de la carte pour le jeu de données simulées.

Longueur moyenne d'intervalle	intSOM-DMahalanobis	intSOM-CMahalanobis	intSOM-L_2
$\boldsymbol{L = 1.5}$	tpe : 0.39% (0.082%) $OERC$: 10.57% (0.73%)	tpe : 0.43% (0.084%) $OERC$: 18.55% (0.82%)	tpe 0.44% (0.099%) $OERC$: 42.95% (1.65%)
$\boldsymbol{L = 3}$	tpe : 0.43% (0.079%) $OERC$: 17.55% (1.08%)	tpe : 0.48% (0.037%) $OERC$: 20.35% (1.00%)	tpe : 0.40% (0.11%) $OERC$: 44.02% (2.02%)
$\boldsymbol{L = 4}$	tpe : 0.42% (0.085%) $OERC$: 24.54% (4.50%)	tpe : 0.41% (0.10%) $OERC$: 34.85% (10.12%)	tpe : 0.40% (0.12%) $OERC$: 44.12% (2.12%)

Les résultats obtenus montrent que la méthode intSOM-DMahalanobis permet d'obtenir une meilleure classification pour tous les jeux de données simulées, suivie par la méthode intSOM-CMahalanobis qui est meilleure que la méthode intSOM-L_2 où seulement des classes de forme sphérique sont reconnues. Les trois méthodes permettent d'obtenir une erreur topographique assez faible ce qui prouve que la topologie est préservée. En effet, dans tous les cas, la carte est bien ordonnée et les classes voisines sont représentées par des neurones voisins.

3.7.2 Jeu de données France-météo

Ce jeu de données comprend la moyenne des températures minimales et la moyenne des températures maximales pour un mois, et ce, pour tous les mois de l'année 2010 (voir tableau A.1, p.186). Le jeu de données est alors constitué de $n = 106$ observations décrites par $p = 12$ variables intervalles. La borne minimale et la borne maximale de chaque intervalle sont respectivement la moyenne des températures minimales et la moyenne des températures maximales enregistrées durant le mois en question.

Apprentissage de la carte. Ce jeu de données est utilisé pour l'apprentissage d'une carte auto-organisatrice de $K = 9$ neurones arrangés suivant une grille carrée. L'apprentissage se fait suivant les trois méthodes : intSOM-DMahalanobis (voir algorithme 3.13, p.116), intSOM-CMahalanobis (voir algorithme 3.12, p.114) et intSOM-L_2 (voir algorithme 3.11+distance L_2).
Pour les trois méthodes, le nombre total d'itérations est $T = 600$, le nombre d'itérations de la première phase pour la méthode intSOM-DMahalanobis est $T1 = 540$, les valeurs initiales et finales du rayon de voisinage sont $\sigma_{init} = 3$ et $\sigma_{final} = 0.1$. La partition initiale est choisie au hasard ce qui fait que les vecteurs prototypes initiaux sont aléatoires et peuvent conduire à des résultats différents à chaque fois. Pour cela, chaque algorithme est lancé 20 fois et les résultats de la lancée qui minimise le plus le critère G_{dyn} (voir équation (3.33), p.114) sont adoptés.

Résultats obtenus par la méthode intSOM-DMahalanobis. L'apprentissage de la carte nous mène à une partition de 9 classes. La figure 3.7 montre la répartition des observations sur les 9 neurones, et la figure 3.8 représente la carte

géographique de la France contenant les 106 stations météorologiques regroupées en classes compte tenu de leur répartition sur les 9 neurones.

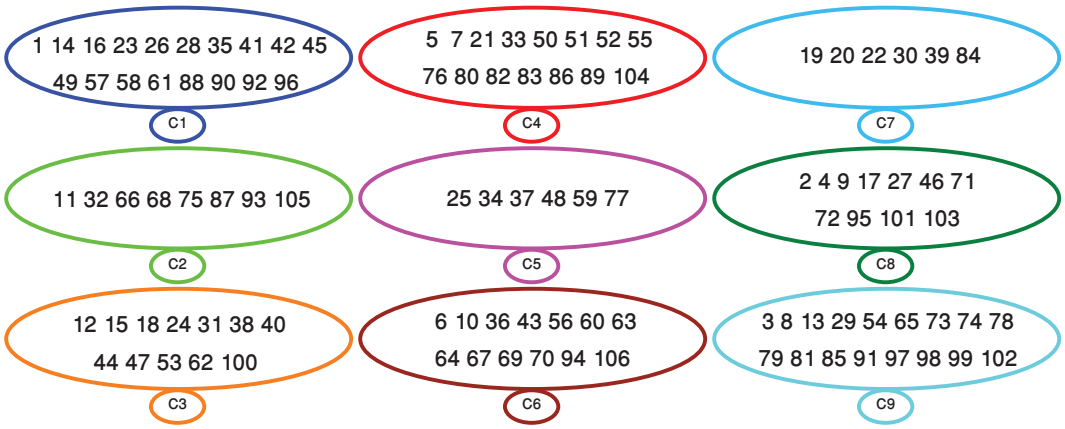


FIGURE 3.7 – Répartition des stations françaises sur les 9 neurones avec la méthode intSOM-DMahalanobis.

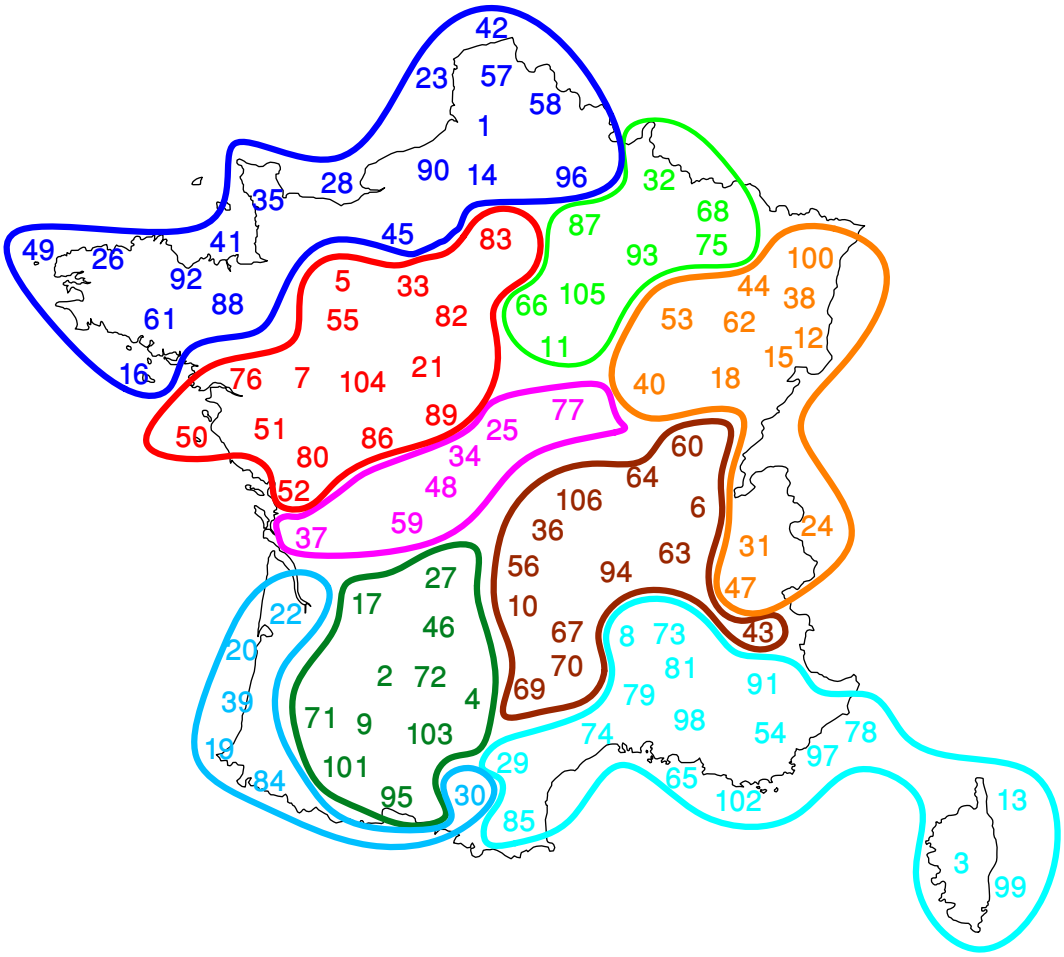


FIGURE 3.8 – Répartition des classes sur la carte géographique de la France avec la méthode intSOM-DMahalanobis.

Les résultats montrent que les stations installées dans des zones proches géographiquement sont affectées au même neurone. Nous pourrons imaginer que l'emplacement géographique d'une ville a une forte influence sur son climat. Par exemple, en France, les villes situées au Sud possèdent un climat plus chaud que celles situées dans l'Est ou dans le Nord du pays. D'après la figure 3.7 nous remarquons que le neurone C_1 contient les stations installées au Nord-Ouest du pays, alors que son opposé, le neurone C_9 contient les stations installées dans le Sud-Est du pays. Un déplacement vers le bas et vers la droite sur la carte auto-organisatrice correspond à un déplacement vers l'Est et vers le Sud sur la carte géographique de France. De plus, deux neurons voisins sur la carte auto-organisatrice représentent deux classes voisines sur la carte géographiques ce qui prouve que la topologie est préservée.

L'erreur topographique obtenue est $te = 4.7\%$. La figure 3.9 est le résultat de l'ACP appliquée aux vecteurs prototypes et aux données. Les rectangles correspondant aux vecteurs prototypes sont connectés par leurs centres dont les numéros désignent le neurone correspondant. Les connexions entre les centres correspondent parfaitement à l'arrangement des neurones sur la carte. Par exemple, le vecteur prototype du cinquième neurone est bien connecté aux vecteurs prototypes des neurones 2, 4, 6 et 8. Le tableau 3.3 montre les valeurs prises par les vecteurs prototypes suivant la première et la deuxième composantes principales.

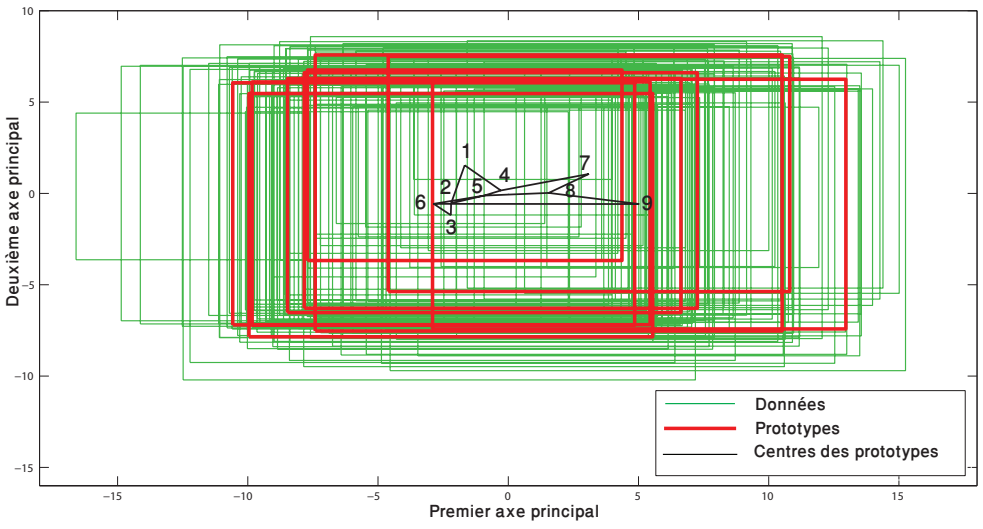


FIGURE 3.9 – ACP des vecteurs prototypes et des données du jeu de données France-météo avec la méthode intSOM-DMahalanobis.

Tableau 3.3 – Vecteurs prototypes du jeu de données France-météo suivant les 2 premières composantes principales avec la méthode intSOM-DMahalanobis.

Neurone	1ère composante principale	2ème composante principale
1	[-7.71,4.37]	[-3.68,6.75]
2	[-9.84,5.45]	[-7.19,6.10]
3	[-9.95,5.55]	[-7.85,5.47]
4	[-7.83,7.26]	[-6.29,6.61]
5	[-8.48,6.63]	[-6.53,6.29]
6	[-10.58,4.85]	[-7.19,6.05]
7	[-4.60,10.80]	[-5.37,7.45]
8	[-7.41,10.51]	[-7.53,7.58]
9	[-2.90,12.96]	[-7.42,6.25]

Résultats obtenus par la méthode intSOM-CMahalanobis. L'apprentissage de la carte nous mène à une partition de 9 classes. La figure 3.10 montre la répartition des observations sur les 9 neurones et la figure 3.11 représente la carte géographique de la France contenant les 106 stations météorologiques regroupées en classes compte tenu de leur répartition sur les 9 neurones.

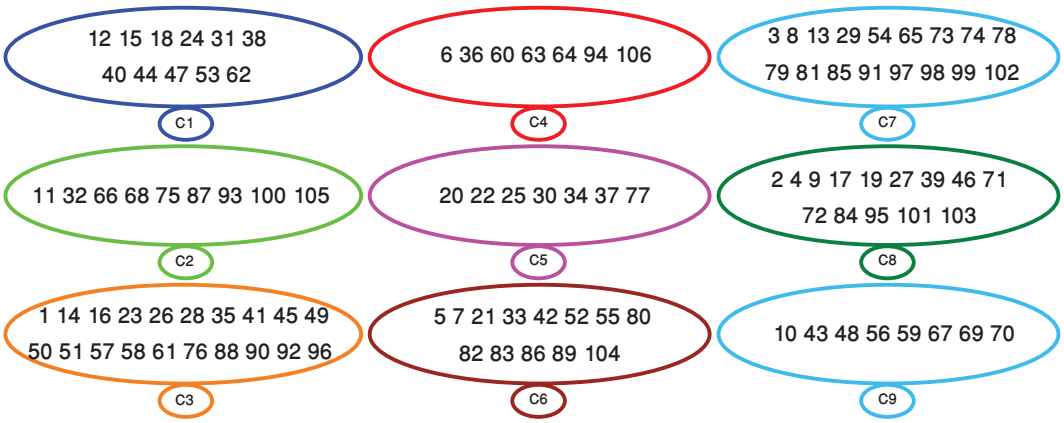


FIGURE 3.10 – Répartition des stations françaises sur les 9 neurones avec la méthode intSOM-CMahalanobis.

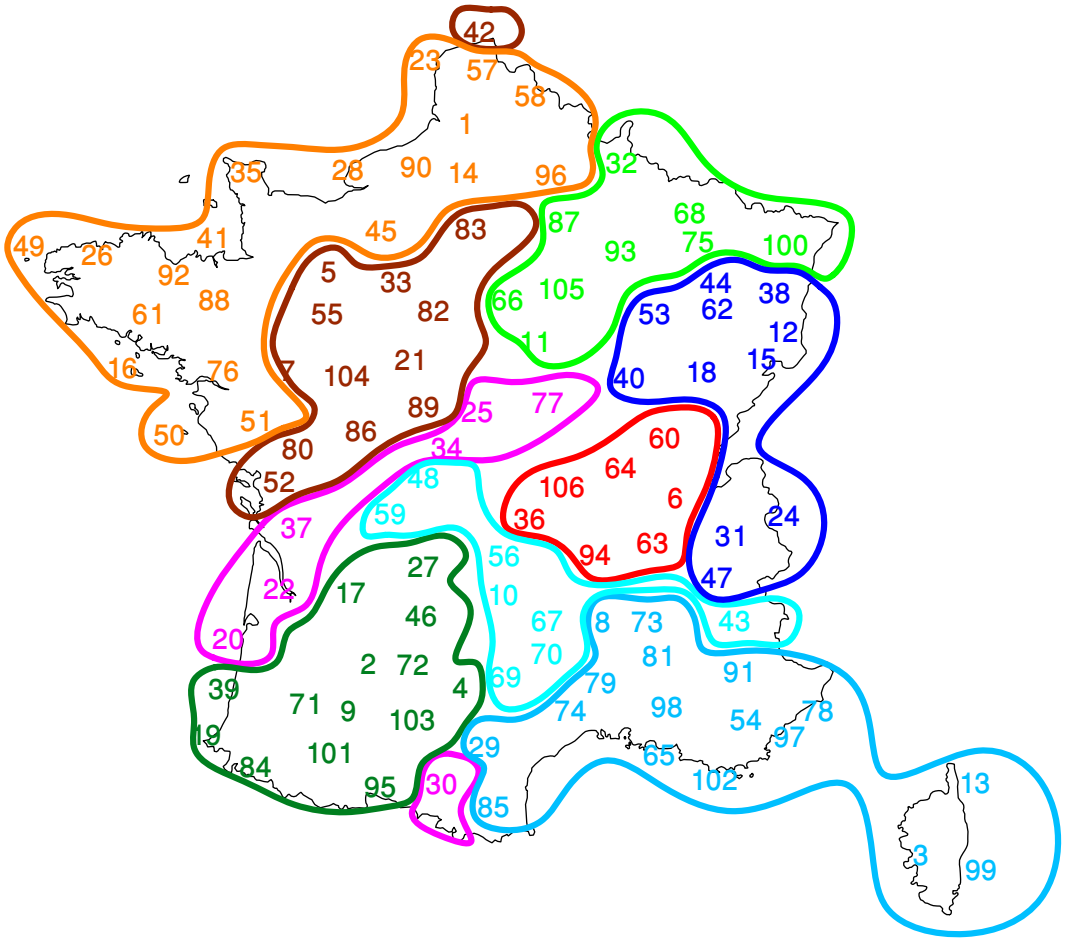


FIGURE 3.11 – Répartition des classes sur la carte géographique de la France avec la méthode intSOM-CMahalanobis.

Les résultats montrent que les stations installées dans des zones proches géographiquement sont affectées au même neurone, à l'exception des stations 42, 30 et 43. L'erreur topographique obtenue est $te = 6.6\%$. La figure 3.12 est le résultat de l'ACP appliquée aux vecteurs prototypes et aux données. Nous remarquons que les connexions des centres des rectangles prototypes respectent l'arrangement des neurones sur la carte auto-organisatrice. Le tableau 3.4 montre les valeurs prises par les vecteurs prototypes suivant la première et la deuxième composantes principales.

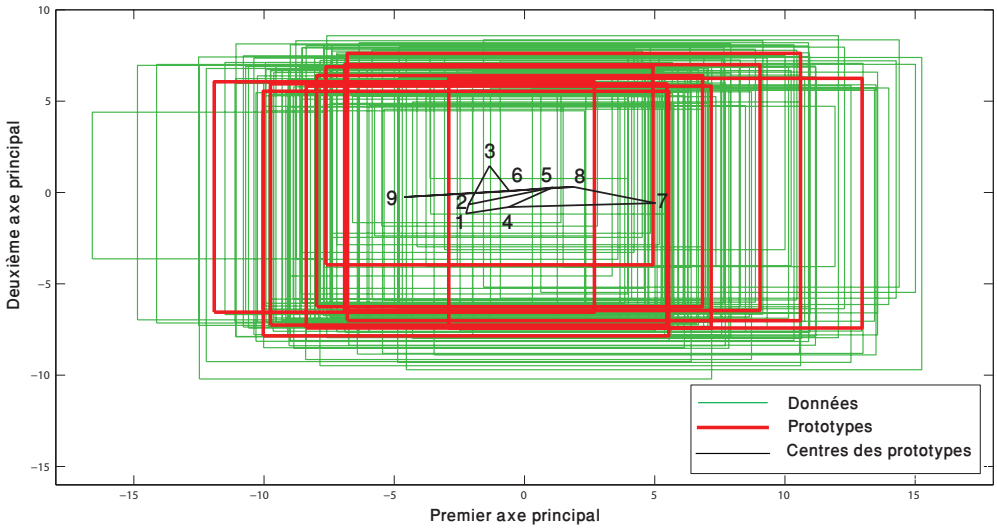


FIGURE 3.12 – ACP des vecteurs prototypes et des données du jeu de données France-météo avec la méthode intSOM-CMahalanobis.

Tableau 3.4 – Vecteurs prototypes du jeu de données France-météo suivant les premières composantes principales avec la méthode intSOM-CMahalanobis.

Neurone	1ère composante principale	2ème composante principale
1	[-10.03,5.54]	[-7.85,5.54]
2	[-9.75,5.47]	[-7.26,5.94]
3	[-7.62,4.93]	[-3.96,6.88]
4	[-8.38,7.17]	[-7.42,5.82]
5	[-6.90,9.05]	[-6.46,6.99]
6	[-7.98,6.83]	[-6.25,6.40]
7	[-2.90,12.96]	[-7.42,6.25]
8	[-6.80,10.60]	[-7.02,7.61]
9	[-11.91,2.68]	[-6.56,6.06]

Résultats obtenus par la méthode intSOM-L_2. L'apprentissage de la carte nous mène à une partition de 9 classes. La figure 3.13 montre la répartition des observations sur les 9 neurones, et la figure 3.14 représente la carte géographique de la France contenant les 106 stations météorologiques regroupées en classes compte tenu de leur répartition sur les 9 neurones.

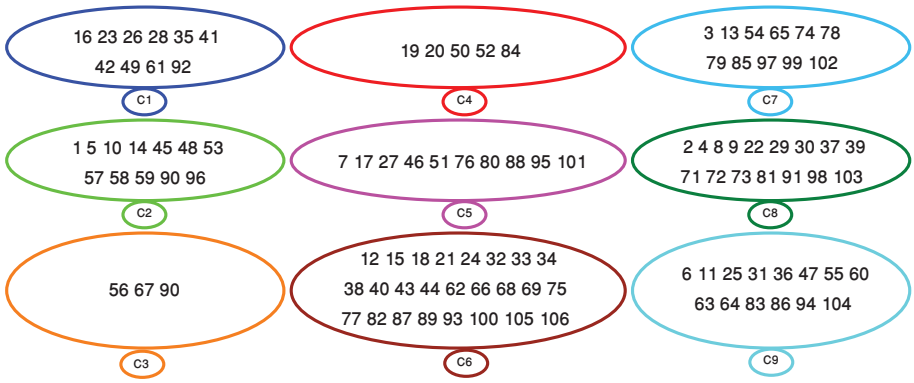


FIGURE 3.13 – Répartition des stations françaises sur les 9 neurones avec la méthode intSOM-L_2.

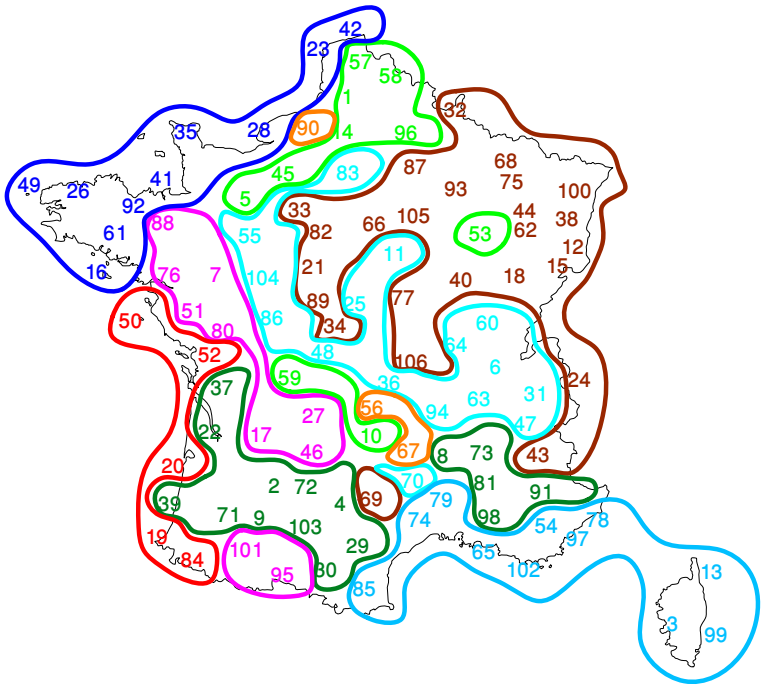


FIGURE 3.14 – Répartition des classes sur la carte géographique de la France avec la méthode intSOM-L_2.

Avec cette méthode, seulement les neurones C_1, C_3, C_4 et C_7 contiennent des stations proches géographiquement. L'erreur topographique est $te = 6.6\%$. La figure 3.15 est le résultat de l'ACP appliquée aux vecteurs prototypes et aux données. Nous remarquons que les connexions des centres des rectangles prototypes respectent bien l'arrangement des neurones sur la carte auto-organisatrice. Le tableau 3.5 montre

les valeurs prises par les vecteurs prototypes suivant la première et la deuxième composantes principales.

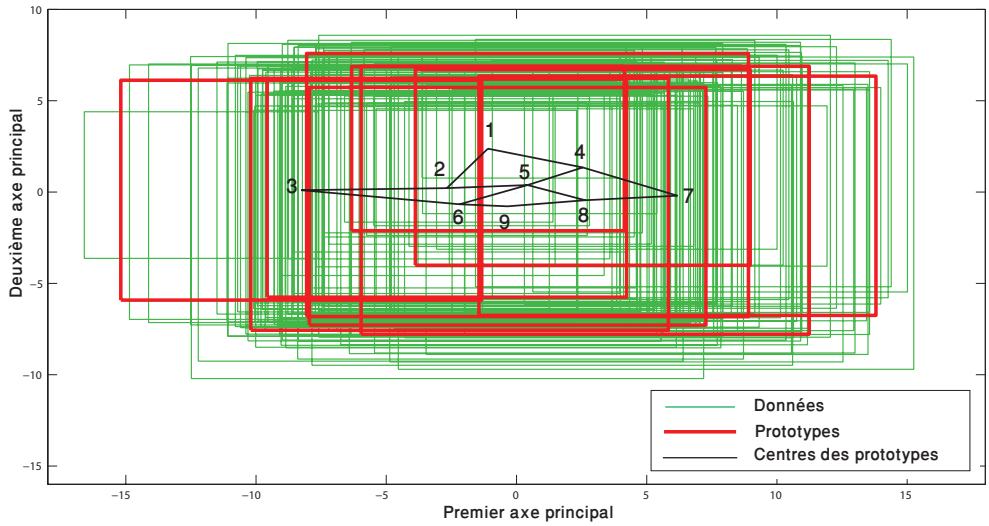


FIGURE 3.15 – ACP des vecteurs prototypes et des données du jeu de données France-météo avec la méthode intSOM-L_2.

Tableau 3.5 – Vecteurs prototypes du jeu de données France-météo suivant les 2 premières composantes principales avec la méthode intSOM-L_2.

Neurone	**1ère composante principale**	**2ème composante principale**
1	[-6.33,4.14]	[-2.12,6.87]
2	[-9.57,4.23]	[-5.75,6.18]
3	[-15.19,-1.33]	[-5.92,6.12]
4	[-3.88,8.97]	[-4.00,6.70]
5	[-8.08,8.91]	[-6.82,7.58]
6	[-10.20,5.83]	[-7.57,6.22]
7	[-1.44,13.80]	[-6.75,6.34]
8	[-5.96,11.24]	[-7.78,6.88]
9	[-7.95,7.27]	[-7.29,5.72]

Comparaison entre les trois méthodes. Afin de comparer les trois méthodes, nous proposons de calculer la distance géographique entre deux stations en utilisant leurs latitude et longitude comme suit :

$$d_{(s_1,s_2)} = \sqrt{(lat_1 - lat_2)^2 + (long_1 - long_2)^2} \tag{3.35}$$

Pour chaque classe C_k, nous calculons la moyenne des distances séparant chaque station du centre de gravité g_k selon :

$$D_{C_k} = \frac{\sum_{s_i \in C_k} d_{(s_i,g_k)}}{|C_k|} \tag{3.36}$$

où $|C_k|$ est le nombre de stations de la classe C_k.

Le tableau 3.6 montre la distance moyenne par classe pour les trois méthodes.

Tableau 3.6 – Distance géographique moyenne par classe pour le jeu de données France-météo.

Classes	**intSOM-DMahalanobis**	**intSOM-CMahalanobis**	**intSOM-L_2**
C_1	2.59	1.19	2.09
C_2	1.18	1.39	2.10
C_3	1.23	2.55	1.89
C_4	1.45	0.88	2.32
C_5	1.02	2.02	1.92
C_6	1.28	1.40	1.75
C_7	1.25	1.83	0.36
C_8	0.89	1.00	2.10
C_9	1.83	1.36	2.27
Total	**12.72**	**13.62**	**16.80**

La méthode intSOM-DMahalanobis permet d'obtenir la plus petite distance moyenne totale.

3.7.3 Jeu de données Liban-météo

Ce jeu de données intervalles représente des températures collectées dans 42 stations météorologiques libanaises durant les 12 mois de l'année 2010 (voir tableau A.2, p.188). Chaque observation ou station est décrite par 12 variables intervalles représentant les mois. La borne inférieure et la borne supérieure de chaque intervalle sont respectivement la moyenne des températures minimales et la moyenne des températures maximales enregistrées durant le mois en question.

Apprentissage de la carte avec la méthode intSOM-DYN et la distance de Hausdorff

On se propose de construire pour ce jeu de données une carte auto-organisatrice dans le but de classifier les données en préservant leur topologie. La carte auto-organisatrice est constituée de quatre neurones arrangés suivant une grille unidimensionnelle (1x4) dans le but de classifier les stations en 4 zones climatiques qui reflètent la réalité du climat au Liban : la première zone contient les régions du littoral ayant une altitude inférieure à 200m et possédant un climat chaud et humide, la deuxième zone regroupe les régions proches du littoral se trouvant à une altitude plus élevée (entre 200m et 800m), la troisième zone contient les régions montagneuses froides ayant une altitude supérieure à 1000m, et la quatrième zone contient les régions de la plaine de la Békaa possédant un climat sec plus ou moins désertique.

L'apprentissage se fait suivant la méthode intSOM-DYN (voir algorithme 3.10, p.106) associé à une combinaison de type L_1 de distances de Hausdorff, que nous notons par la suite intSOM-DYN-Hausdorff. Cet apprentissage est fait en trois itérations principales, la première correspond à un rayon de voisinage initial égal à $\sigma_{init} = \sigma(1) = 2$, la seconde à un rayon de voisinage égal à $\sigma(2) = 1$ et la troisième à un rayon de voisinage final égal à $\sigma_{final} = \sigma(3) = 0.1$ (du fait qu'une valeur nulle rend le calcul de la fonction de voisinage impossible). Les vecteurs prototypes initiaux sont choisis aléatoirement dans l'ensemble des observations, ce qui pourrait influencer les résultats obtenus. Pour cette raison, 20 lancées de cet algorithme sont exécutées avec à chaque fois une initialisation différente pour les vecteurs prototypes. Les résultats produits par la lancée qui permet de minimiser le plus le critère G_{intSOM} (voir équation (3.11), p.103) sont retenus. La première, deuxième et troisième itération engendrent respectivement 5, 4 et 5 sous-itérations pour minimiser le critère G_{intSOM} (voir équation (3.11), p.103) où la distance d est la distance de Hausdorff-L_1.

Résultats. En fin d'apprentissage, nous obtenons une partition de 4 classes. La figure 3.16 montre la répartition des stations sur les quatre neurones de la carte. La figure 3.17 représente la carte géographique du Liban contenant les stations avec les classes qu'elles forment. Le premier neurone contient les stations installées dans la plaine de la Békaa située dans l'Est et le Nord-Est du pays. Les stations

installées dans les montagnes sont affectées au deuxième neurone et les stations installées dans le Sud et dans des villes ou villages proches du littoral sont affectées au troisième neurone. Au quatrième neurone sont affectées des stations installées le long du littoral libanais. La figure 3.18 est le résultat de l'ACP appliquée aux vecteurs de données et aux vecteurs prototypes. Le tableau 3.7 montre les valeurs prises par les vecteurs prototypes suivant la première et deuxième composantes principales.

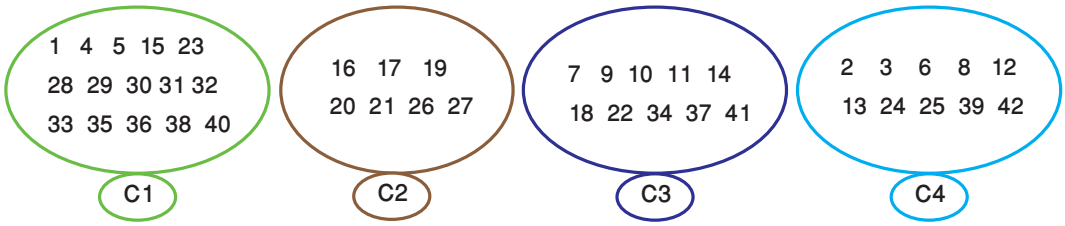


FIGURE 3.16 – Répartition des stations du jeu de données Liban-météo sur les 4 neurones avec la méthode intSOM-DYN-Hausdorff.

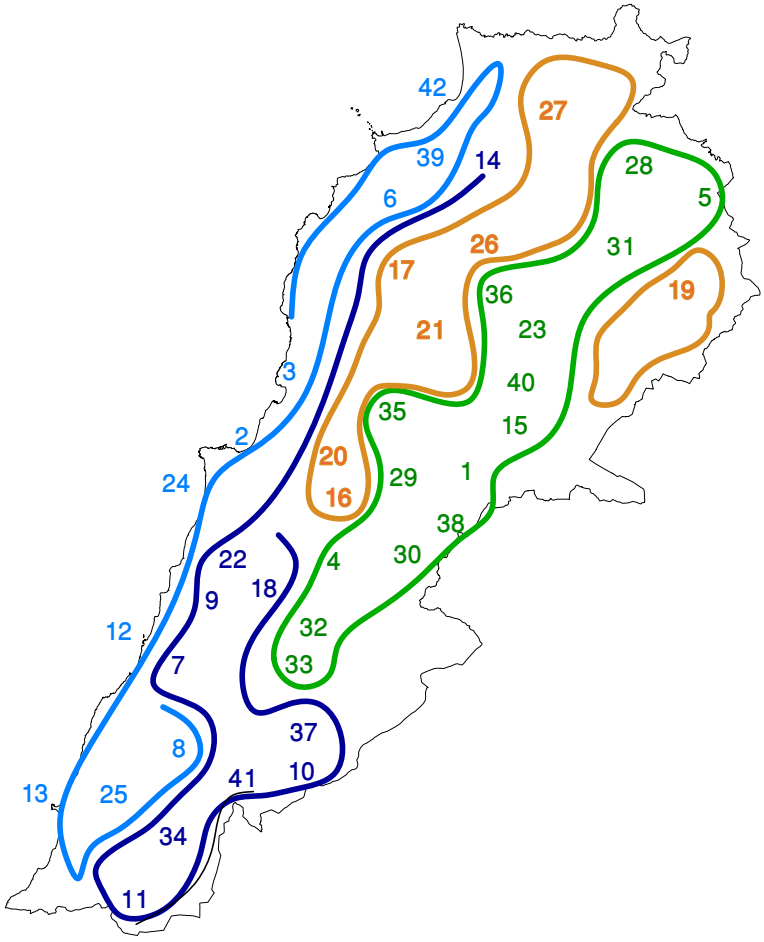


FIGURE 3.17 – Répartition des classes du jeu de données Liban-météo sur la carte géographique du Liban avec la méthode intSOM-DYN-Hausdorff.

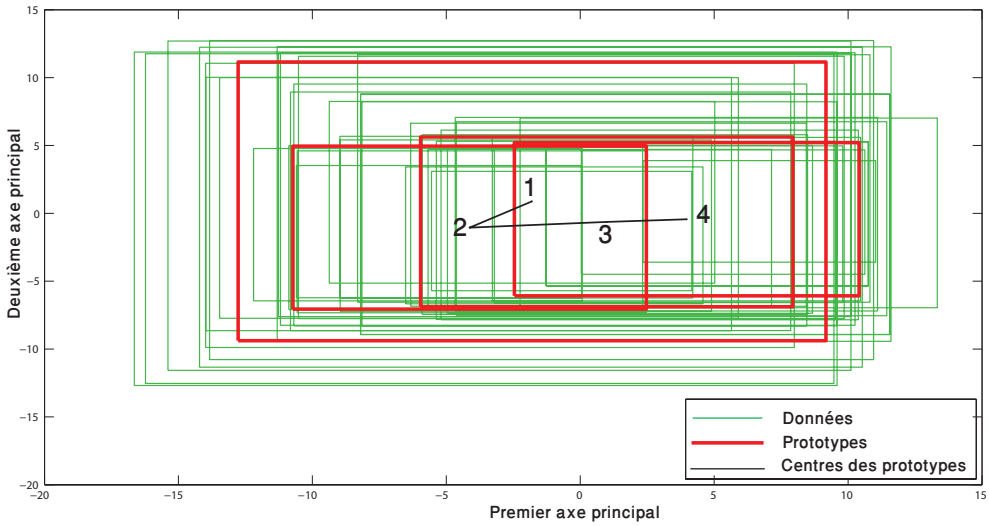


FIGURE 3.18 – ACP des vecteurs prototypes et des données du jeu de données Liban-météo avec la méthode intSOM-DYN-Hausdorff.

Tableau 3.7 – Vecteurs prototypes du jeu de données Liban-météo suivant les 2 premières composantes principales avec la méthode intSOM-DYN-Hausdorff.

Neurone	**1ère composante principale**	**2ème composante principale**
1	[-12.77,9.17]	[-9.38,11.15]
2	[-10.73,2.47]	[-7.056,4.95]
3	[-5.96,7.94]	[-6.88,5.62]
4	[-2.46,10.41]	[-6.08,5.21]

Comparaison avec d'autres méthodes

En guise de comparaison, nous avons classifié ce jeu de données avec la méthode des nuées dynamiques associée à la distance de Hausdorff (Chavent et Lechevallier, 2002). Le tableau donne les résultats de classification obtenus sachant que l'algorithme a été lancé 20 fois avec à chaque fois une initialisation différente pour les représentants des classes et seulement le résultat de la lancée permettant de minimiser le plus le critère des nuées dynamiques est retenu.

Tableau 3.8 – Résultats de la classification pour le jeu de données Liban-météo avec l'algorithme des nuées dynamiques associé à la distance de Hausdorff.

Classe	**Observations**
C_1	16 17 19 20 21 22 26 27 36
C_2	2 3 12 13 24 25 42
C_3	6 7 8 9 10 11 14 18 23 28 32 33 34 37 39 41
C_4	1 4 5 15 29 30 31 35 38 40

La classe C_1 contient les stations installées plutôt dans des régions à haute altitude, la classe C_2 contient des stations installées sur le littoral. La classe C_3 contient des stations installées dans le Sud et dans des régions proches du littoral mais elle contient aussi des stations installées dans la plaine de la Békaa comme les stations 28, 32 et 33. La totalité des stations de la classe C_4 contient des stations installées dans la plaine de la Békaa. Dans le souci de mieux comparer ces deux méthodes, et du fait que l'altitude d'une ville ou d'un village au Liban renseigne en quelques sortes sur son climat, nous avons calculé la moyenne et l'écart-type des altitudes de chaque classe obtenus avec les deux méthodes tout en exposant les résultats dans le tableau 3.9.

Tableau 3.9 – Moyenne et écart-type des altitudes par classe pour le jeu de données Liban-météo.

Classe	intSOM-DYN-Hausdorff		**Nuées dynamiques**	
	Moyenne	**Écart-type**	**Moyenne**	**Écart-type**
C_1	955.30 m	152.63 m	1294.4 m	205.72 m
C_2	1365.70 m	155.33 m	152.10 m	164.46 m
C_3	706.00 m	179.84 m	682.80 m	257.26 m
C_4	204.00 m	167.09 m	935 m	161.59 m
Total des écart-types		657.9 m	**Total des écart-types**	789.04 m

D'après le tableau nous pouvons conclure que les classes obtenues avec la méthode des nuées dynamiques contiennent des stations présentant un plus grand écart d'altitude qu'avec la méthode intSOM-DYN-Hausdorff. Donc, la méthode intSOM-DYN-Hausdorff donne un résultat plus réaliste du climat au Liban que la méthode des nuées dynamiques. De plus, la méthode intSOM-DYN-Hausdorff préserve la topologie des données en fournissant une information supplémentaire concernant la proximité des classes.

3.7.4 Jeu de données *Car models*

Le jeu de données *Car models* (De Carvalho *et al.*, 2006) regroupe des informations sur 33 modèles de voitures décrits par 8 variables de type intervalle. Une partition *a priori* de ce jeu de données classifie ces modèles en 4 classes (voir tableau 3.10, p.139).

Apprentissage de la carte avec la méthode intSOM-DYN et la distance L_1

Nous nous proposons de bâtir pour ce jeu de données une carte auto-organisatrice formée de 4 neurones disposés suivant une grille carrée. Du fait que les échelles de mesure des variables sont bien différentes (par exemple : prix et vitesse), une normalisation des données est pratiquée suivant la technique décrite dans la sous-section 3.4.1. L'apprentissage de la carte se fait suivant la méthode intSOM-DYN (voir algorithme 3.10, p.106) associée à la distance L_1 que nous notons par la suite intSOM-DYN-L_1. Cet apprentissage est fait en deux itérations qui correspondent à un rayon de voisinage initial égal à 1 et final égal à 0.1. Les vecteurs prototypes initiaux sont choisis aléatoirement dans l'ensemble des observations, ce qui pourrait influencer les résultats obtenus. Pour cela, 20 lancées de cet algorithme sont exécutées avec à chaque fois une initialisation différente des vecteurs référents, et les résultats produits par la lancée qui permet de minimiser le plus le critère G_{intSOM} (voir équation (3.11), p.103) sont adoptés. Quand le rayon de voisinage est fixé à 1, l'optimisation du critère G_{intSOM} nécessite 4 sous-itérations et quand le rayon de voisinage est réduit à 0.1, l'optimisation du critère G_{intSOM} nécessite 7 sous-itérations.

Tableau 3.10 – Partition a priori du jeu de données *Car models*.

Classe	Modèle
Utilitaire	-Alfa 145/U -Audi A3/U -Punto/U -Fiesta/U -Lancia Y/U -Nissan Micra/U -Corsa/U -Twingo/U -Rover 25/U -Skoda Fabia/U
Berline	-Alfa 156/B -Audi A6/B -BMW serie 3/B -Focus/B -Mercedes Classe C/B -Vectra/B -Rover 75/B -Skoda Octavia/B
Sport	-Aston Martin/S -Ferrari/S -Honda NSK/S -Lamborghini/S -MaseratiGT/S -Mercedes SL/S -Porsche/S
Luxe	-Alfa 166/L -Audi A8/L -BMW serie 5/L -BMW serie 7/L -Lancia K/L -Mercedes Classe E/L -Mercedes Classe S/L -Passat/L

Résultats. La partition finale du jeu de données est illustrée dans la figure 3.19. Tous les modèles de la classe Luxe sont affectés au premier neurone C_1 de la carte. La plupart des modèles de la classe Sport sont affectés au deuxième neurone C_2 de la carte, sauf le modèle Ferrari/S qui appartient au neurone C_1 voisin du neurone C_2. La majorité des modèles de la classe Berline sont affectés au troisième neurone C_3 sauf les deux modèles Audi A6/B et Rover 75/B qui appartiennent au neurone C_1 voisin du neurone C_3. Presque tous les modèles de la classe Utilitaire sont affectés au neurone C_4 à l'exception du modèle Audi A3/U qui appartient au neurone C_2 voisin du neurone C_4. Ceci nous amène à conclure que seulement 4 modèles parmi les 33 sont mal classifiés. La figure 3.20 montre la projection des vecteurs de données et des vecteurs prototypes sur un espace bidimensionnel formé par les deux premiers axes principaux obtenus comme résultat de l'ACP. Nous remarquons que chaque vecteur prototype est bien connecté à ses voisins, ce qui assure la préservation de la topologie.

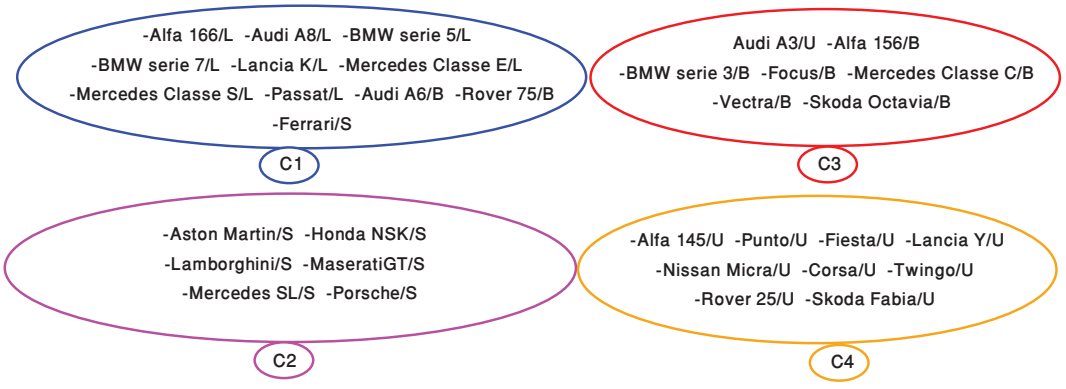


FIGURE 3.19 – Répartition des modèles de voitures sur les 4 neurones de la carte.

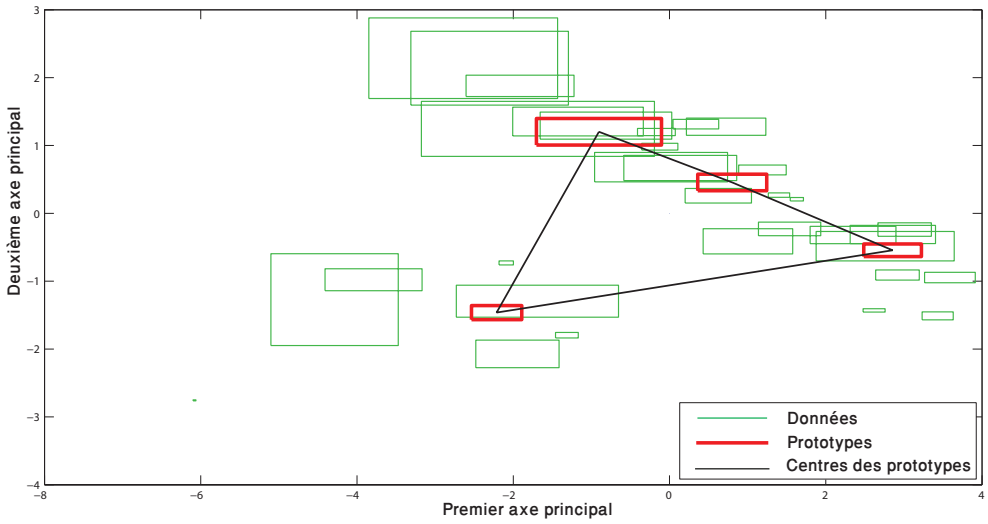


FIGURE 3.20 – ACP des vecteurs prototypes et des données du jeu de données *Car models* avec la méthode intSOM-DYN-L_1.

Dans le but de donner des résultats de classification plus précis, nous avons évalué les résultats de la classification avec les deux critères d'évaluation externes d'une partition : l'indice de Rand corrigé *(CR Index)* et le taux global d'erreur de classification *(OERC)* (voir sous-section 1.4.1, p.39) qui donne le pourcentage des vecteurs de données mal classifiés. Nous avons obtenu pour l'indice de Rand corrigé une valeur de 0.693 et pour le taux global d'erreur de classification la valeur 12.12%. Les mêmes résultats sont obtenus avec la méthode intSOM (voir algorithme 3.11, p.109) associé à une combinaison de type L_1 de distances de Hausdorff, quand les paramètres d'apprentissage choisis sont : rayon initial : $\sigma_{init} = 1$, rayon final : $\sigma_{final} = 0.1$, nombre total d'itérations : $T = 100$.

Comparaison avec d'autres méthodes

Les deux méthodes intSOM-DYN-Hausdorff (voir algorithme 3.10, p.106 avec une combinaison de type L_1 de distances de Hausdorff) et intSOM-DYN-L_2 (voir algorithme 3.10, p.106 avec le carré de la distance L_2), sont utilisées pour l'apprentissage de la carte auto-organisatrice en vue de classifier ce jeu de données en 4 classes en adoptant les même paramètres d'apprentissage utilisés dans la méthode intSOM-DYN-L_1. Les techniques de normalisation, décrites dans les sous-sections

3.4.3 et 3.4.2, sont respectivement utilisées pour normaliser les données avant leur classification avec les méthodes intSOM-DYN-Hausdorff et intSOM-DYN-L_2. Nous comparons aussi les approches proposées à la méthode IABSOM-$L1$ qui consiste à étendre l'algorithme d'optimisation en mode différé des cartes topologiques aux données de type intervalle, en utilisant une distance adaptative entre vecteurs d'intervalles basée sur la distance city-block (De Carvalho *et al.*, 2012). Le tableau 3.11 donne le *CR Index* et le *OERC* des différentes méthodes pour le jeu de données *Car models*.

Tableau 3.11 – Évaluation de la partition obtenue du jeu de données *Car models*.

Méthode	***CR Index***	***OERC***
intSOM-DYN-L_1	0.693	12.12%
intSOM-DYN-Hausdorff	0.693	12.12%
intSOM-DYN-L_2	0.648	15.15%
IABSOM-L_1	0.477	15.20%

Les méthodes intSOM-DYN-L_1 et intSOM-DYN-Hausdorff permettent une meilleure classification de ce jeu de données donnant le plus haut *CR Index* et le plus bas pourcentage d'observations mal classifiées *OERC*.

Sensibilité des méthodes proposées face aux points aberrants. Afin de tester la robustesse des méthodes proposées, nous avons introduit 4 observations aberrantes au jeu de données *Car models*, en prenant de chaque classe *a priori* une observation et en multipliant ses valeurs par 100. Le tableau 3.12 donne le *CR Index* et le *OERC* des différentes méthodes pour le jeu de données *Car models* après avoir introduit les observations aberrantes, sachant que les paramètres d'apprentissage ainsi que la normalisation des données sont appliqués d'une façon semblable à ceux du jeu de données *Car models* original.

Tableau 3.12 – Évaluation de la partition obtenue du jeu de données *Car models* avec observations aberrantes.

Méthode	***CR Index***	***OERC***
intSOM-DYN-L_1	0.526	29.72%
intSOM-DYN-Hausdorff	0.477	32.43%
intSOM-DYN-L_2	-0.010	-

D'après les deux tableaux 3.12 et 3.11, nous remarquons que les méthodes intSOM-DYN-L_1 et intSOM-DYN-Hausdorff sont plus robustes que la méthode intSOM-DYN-L_2 face à la présence d'observations aberrantes. Ceci s'explique par le fait que le calcul des vecteurs prototypes dans le cas des distances L_1 et Hausdorff se base sur la médiane plutôt que sur la moyenne. Avec la méthode intSOM-DYN-L_2, les observations sont affectées à deux neurones parmi les quatre, ce qui rend le calcul du $OERC$ impossible.

3.8 Conclusion

Vu l'importance des données de type intervalle pour représenter les mesures issues des applications réelles, nous avons proposé dans ce chapitre plusieurs approches pour classifier les données intervalles avec les cartes auto-organisatrices. Ces approches se divisent en deux familles principales. Dans la première famille, l'apprentissage de la carte se fait en mode différé mais en optimisant un critère, donc, l'utilisateur ne doit plus fournir le nombre d'itérations à l'algorithme. Dans la deuxième famille, l'apprentissage de la carte auto-organisatrice est réalisé d'une façon heuristique toujours en mode différé. Ce qui distingue ces différents algorithmes c'est la distance utilisée qui engendre à chaque fois une règle différente pour la recherche du neurone vainqueur et une règle différente pour la mise à jour des vecteurs prototypes. Nous avons proposé aussi deux algorithmes d'apprentissage adaptatif, toujours en mode différé, basés sur la distance de Mahalanobis qui permet de prendre en compte la corrélation entre les variables ainsi que leurs variabilités, conduisant alors à une classification adaptative avec des classes de formes ellipsoïdales. De plus, les expériences menées ont montré, comme attendu, que l'utilisation de la distance L_1 ou d'une distance basée sur la distance de Hausdorff rend la classification plus robuste face à la présence d'observations aberrantes. Les résultats expérimentaux obtenus, ainsi que les résultats de comparaison avec d'autres méthodes de classification de données intervalles, prouvent la validité des approches proposées.

Dans le chapitre 4, nous allons utiliser les algorithmes proposés dans ce chapitre pour bâtir **une carte auto-organisatrice pour les données discrétisées *(binned data)* dans le but d'accélérer l'apprentissage des cartes classiques.**

Chapitre 4

Cartes auto-organisatrices pour les données discrétisées

4.1 Introduction

Les cartes auto-organisatrices pour les données ponctuelles sont connues par leur capacité à traiter de larges jeux de données, vu la complexité linéaire en fonction du nombre d'observations de leur algorithme d'apprentissage. Toutefois, des problèmes concernant le temps d'apprentissage s'imposent quand la taille de l'échantillon est importante. De plus, la création d'une carte auto-organisatrice pour un grand jeu de données pourrait présenter des problèmes d'allocation mémoire. Plusieurs solutions ont été proposées dans la littérature pour réduire le temps d'apprentissage d'une carte auto-organisatrice : Koikkalainen (1995b) a proposé le *TS-SOM*, un algorithme qui consiste à accélérer la recherche du neurone vainqueur. Su et H.T. (2000) ont présenté une approche efficace pour créer une carte auto-organisatrice en trois étapes : la première étape consiste à sélectionner les centres des classes en utilisant l'algorithme des centres-mobiles, ces centres sont distribués d'une manière heuristique suivant une grille rectangulaire lors d'une deuxième étape et le réglage final de la carte est réalisé lors d'une troisième étape moyennant l'algorithme d'apprentissage standard des cartes auto-organisatrices. Fiannaca *et al.* (2011) ont proposé une application de la méthode du recuit simulé *(simulated annealing)* dans le but d'accélérer le mécanisme d'apprentissage des cartes auto-organisatrices .

La solution que nous proposons dans ce chapitre, pour réduire le temps d'ap-

prentissage et pour contourner les problèmes d'allocation mémoire, est de discrétiser les données en les regroupant en *bins* de façon à en créer un histogramme multidimensionnel. Ce sont les *bins* qui seront présentés à la carte pour l'entraîner à la place des observations originales. Les données discrétisées jouent un rôle important dans le traitement des données massives (Poosala, 1997; Matias *et al.*, 1998; Lee *et al.*, 1999) car elles permettent de réduire considérablement les données à traiter. Nous citons dans ce qui suit quelques travaux effectués dans la classification de données discrétisées.

L'application de l'algorithme EM *(Expectation-Maximization)* aux données discrétisées a été introduite de façon très générale par Dempster *et al.* (1977) où on considère ce problème comme une situation particulière de données manquantes. Cette approche a été développée par la suite dans le cas monodimensionnel pour le modèle de mélange par McLachlan et Jones (1988) puis généralisée par Cadez *et al.* (2002) dans le cas multidimensionnel. Dans l'approche de Cadez *et al.* (2002), la résolution du problème d'estimation des paramètres du modèle de mélange (approche mélange), connaissant seulement les données discrétisées, s'effectue en maximisant la vraisemblance par l'algorithme *binned-EM (Binned Expectation-Maximization)*. La résolution du problème d'estimation simultanée des paramètres du modèle de mélange et de la partition (approche classification), connaissant seulement les données discrétisées, s'effectue en maximisant un critère de vraisemblance complétée, par l'algorithme bin-EM-CEM (Samé, 2004; Samé *et al.*, 2006). Pour garantir un temps de calcul constant et, en même temps, obtenir une précision de résultats raisonnable, Hamdan et Govaert (2004b) ont développé une méthode originale de discrétisation de données incertaines permettant de prendre en compte l'incertitude ou l'imprécision de données afin d'améliorer les résultats fournis par les algorithmes de classification de données discrétisées. Lorsque cette méthode est utilisée avec les algorithmes *binned-EM* et bin-EM-CEM, elle permet de se rapprocher des résultats fournis par les algorithmes EM et CEM respectivement (Hamdan et Govaert, 2004b; Hamdan, 2005, 2006).

Pour combiner les avantages de l'utilisation des modèles de mélange gaussiens parcimonieux et des données discrétisées, quatorze algorithmes *binned-EM* ont été développés pour les modèles de mélange gaussiens parcimonieux : deux algorithmes pour les modèles sphériques (Hamdan et Wu, 2011), quatre algorithmes pour les modèles diagonales (Wu et Hamdan, 2011) et huit algorithmes pour les modèles gaussiens parcimonieux les plus généraux (Wu et Hamdan, 2012). Quatorze algo-

rithmes *bin-EM-CEM* ont été également développés pour les modèles de mélange gaussiens parcimonieux (Hamdan et Wu, 2013a; Wu et Hamdan, 2013a). Pour automatiser le choix d'un modèle parmi ces quatorze modèles ainsi que le choix d'un nombre de classes pour la classification de données discrétisées, il est commode d'utiliser le critère BIC dans le cas de l'approche mélange (Wu et Hamdan, 2013b) et le critère ICL dans le cas de l'approche classification (Hamdan et Wu, 2013b).

Dans ce chapitre, nous allons proposer un algorithme pour la classification des données discrétisées moyennant les cartes auto-organisatrices. Tout d'abord, nous donnons une définition des données discrétisées. Ensuite, nous exposons en détail la méthode de classification que nous proposons pour classifier ce type de données. Avant de conclure, nous présentons les résultats des expériences menées sur des jeux de données simulées ainsi que les résultats obtenus de la segmentation d'images en utilisant les méthodes proposées.

4.2 Données discrétisées

La discrétisation de données consiste à partitionner l'espace des observations en des régions disjointes, chaque région étant reconnue par sa fréquence qui est le nombre d'observations qu'elle contient. Dans ce chapitre, nous proposons de faire une discrétisation uniforme des données où les régions sont des hypercubes qui, associés à leurs fréquences, forment un histogramme multidimensionnel.

Soit Ω un ensemble de n vecteurs $\boldsymbol{x}_i \in \mathbb{R}^p\, i \in \{1, \ldots, n\}$ représentés par des points de l'espace $\mathbb{R}^p$. Nous proposons de créer à partir de ces données un histogramme multidimensionnel formé de *bins*. Cette opération est réalisée en divisant chaque dimension en un nombre d'intervalles de longueur égale, ce qui produit V régions. Un hypercube (cube p-dimensionnel) constitue la base d'un *bin* dont la hauteur est la fréquence des vecteurs de données appartenant à la région correspondante. Dans le cas unidimensionnel, chaque *bin* se réduit à une colonne dont la largeur vaut la longueur de l'intervalle de base et la hauteur vaut la fréquence des observations appartenant à cet intervalle.

4.3 Cartes auto-organisatrices pour les données discrétisées

L'opération de discrétisation d'un ensemble de données p-dimensionnelles produit V *bins*. Chaque *bin* sera noté par $\boldsymbol{\mathcal{H}}_\theta$ avec $\theta \in \{1, \ldots, B\}$, B étant le nombre de *bins* non vides, donc $B \leqslant V$. Le *bin* $\boldsymbol{\mathcal{H}}_\theta$ est représenté par un vecteur de p intervalles noté par $\boldsymbol{\mathcal{R}}_\theta = ([\alpha_\theta^1, \beta_\theta^1], \ldots, [\alpha_\theta^p, \beta_\theta^p])^T$ et une fréquence notée F_θ.

Nous proposons de construire une carte auto-organisatrice pour l'ensemble des *bins* $\{\boldsymbol{\mathcal{H}}_\theta,\ \theta \in \{1, \ldots, B\}\}$, composée de K neurones. Le prototype $\boldsymbol{W}_k$ de chaque neurone k est un vecteur de p intervalles noté $\boldsymbol{W}_k = ([u_k^1, v_k^1], \ldots, [u_k^p, v_k^p])^T$. L'ensemble de tous les prototypes est représenté par la matrice $\mathbb{W}$ de taille $K \times p$.

4.3.1 Apprentissage de la carte en mode différé en optimisant un critère

L'apprentissage de la carte se fait en mode différé en présentant, à chaque itération, tout l'ensemble des *bins* à la carte. À chaque itération, un rayon de voisinage est fixé et plusieurs sous-itérations sont engendrées ayant pour but de minimiser le critère suivant :

$$G_{binSOM}(\mathbb{W}) = \sum_{k=1}^{K} \sum_{\theta=1}^{B} h_{kc_\theta}\big(\sigma(t)\big) F_\theta d(\boldsymbol{\mathcal{R}}_\theta, \boldsymbol{W}_k) \tag{4.1}$$

où :

- $h_{kc_\theta}\big(\sigma(t)\big)$ est la fonction de voisinage gaussienne entre le neurone k et le neurone vainqueur c_θ du *bin* $\boldsymbol{\mathcal{H}}_\theta$.
- $\sigma(t)$ est le rayon de voisinage à l'itération t qui est initialisé à une valeur suffisamment grande en début d'apprentissage pour activer le plus grand nombre de neurones possible, et qui décroît ensuite pour atteindre une valeur presque nulle (seulement le neurone vainqueur est activé).
- F_θ est la fréquence du *bin* $\boldsymbol{\mathcal{H}}_\theta$.
- $\boldsymbol{\mathcal{R}}_\theta$ est le vecteur d'intervalles associé au *bin* $\boldsymbol{\mathcal{H}}_\theta$.
- d est une distance entre vecteurs d'intervalles.

Le rayon de voisinage est initialisée à une grande valeur au début de l'apprentissage (moitié de la plus grande dimension de la carte), puis décroît avec les

itérations pour atteindre une valeur suffisamment faible à la fin de l'apprentissage de façon à activer seulement le neurone vainqueur. Dans les expériences menées dans ce chapitre, nous proposons une décroissance linéaire du rayon de voisinage.

Recherche du neurone vainqueur

Dans le but de minimiser le critère G_{binSOM}, la recherche du neurone vainqueur du *bin* $\boldsymbol{\mathcal{H}}_\theta$ doit se faire tel que (voir sous-section 1.3.3, p.35) :

$$c_\theta = \underset{k\in\{1,\ldots,K\}}{\arg\min} \sum_{\tau=1}^{K} F_\theta h_{k\tau}(t) d\big(\boldsymbol{\mathcal{R}}_\theta, \boldsymbol{\mathcal{W}}_\tau\big) \tag{4.2}$$

Détermination des vecteurs prototypes

Dans le cas où la mesure de proximité d entre vecteurs d'intervalles est le carré de la distance L_2, la minimisation du critère G_{binSOM} requiert la mise à jour suivante des vecteurs prototypes :

$$\boldsymbol{u}_k = \frac{\sum_{\theta=1}^{B} h_{kc_\theta}(t) F_\theta \boldsymbol{a}_\theta}{\sum_{\theta=1}^{B} h_{kc_\theta}(t) F_\theta} \tag{4.3}$$

$$\boldsymbol{v}_k = \frac{\sum_{\theta=1}^{B} h_{kc_\theta}(t) F_\theta \boldsymbol{b}_\theta}{\sum_{\theta=1}^{B} h_{kc_\theta}(t) F_\theta} \tag{4.4}$$

où $\boldsymbol{u}_k = [u_k^1, \ldots, u_k^p]$, $\boldsymbol{a}_\theta = [\alpha_\theta^1, \ldots, \alpha_\theta^p]$, $\boldsymbol{v}_k = [v_k^1, \ldots, v_k^p]$ et $\boldsymbol{b}_\theta = [\beta_\theta^1, \ldots, \beta_\theta^p]$.

Le vecteur prototype d'un neurone k est alors la moyenne pondérée des vecteurs d'intervalles qui représentent les *bins*, chaque poids étant le produit de la fonction de voisinage h_{kc_θ} par la fréquence F_θ du *bin* θ.

L'algorithme 4.14 liste les étapes nécessaires pour réaliser l'apprentissage d'une carte auto-organisatrice en mode différé pour des données discrétisées en optimisant un critère basé sur le carré de la distance L_2.

Algorithme 4.14 Apprentissage en mode différé des cartes auto-organisatrices pour les données discrétisées en optimisant un critère basé sur le carré de la distance L_2 (binSOM-DYN-L_2).

Entrées : Fournir :

- $\boldsymbol{\mathcal{R}}_\theta = \left([\alpha_\theta^1, \beta_\theta^1], \ldots, [\alpha_\theta^p, \beta_\theta^p]\right)$, $\theta \in \{1, \ldots, B\}$ {□ Vecteur d'intervalles associé au *bin* $\boldsymbol{\mathcal{H}}_\theta$}
- F_θ {□ Fréquence associée au *bin* $\boldsymbol{\mathcal{H}}_\theta$}
- $\boldsymbol{\xi}$ {□ Vecteur de dimension n, tel que $\xi(i) = \theta$: Appartenance de l'observation $\boldsymbol{x}_i$ au *bin* $\boldsymbol{\mathcal{H}}_\theta$}
- lig, col {□ Dimensions de la carte, $K = lig \cdot col$}
- $\sigma_{init}, \sigma_{final}$ {□ Valeurs initiale et finale du rayon de voisinage}

Sorties : Récupérer :

- $\mathbb{W}(f)$ {□ Vecteurs prototypes auto-organisés}
- $P_f = \{C_1, \ldots, C_K\}$ {□ Partition finale de l'ensemble des observations}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante associée à la réduction du rayon de voisinage}

$s \leftarrow 0$ {□ s : Itération courante associée à la mise à jour des vecteurs prototypes}

Initialiser les vecteurs prototypes $\mathbb{W}(0)$

Initialiser le rayon de voisinage $\sigma(t)$ à σ_{init}

Apprentissage :

Répéter

Détermination des vecteurs prototypes en optimisant G_{binSOM} (équation (4.1)) :

Répéter

Calcul des valeurs de la fonction de voisinage : $h_{k\tau}(t)$, $k, \tau \in \{1, \ldots, K\}$

Pour $k = 1 \rightarrow K$ **Faire**

Pour $\tau = 1 \rightarrow K$ **Faire**

$h_{k\tau}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_k - \boldsymbol{r}_\tau\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Calcul des neurones vainqueurs des observations :

Pour $\theta = 1 \rightarrow B$ **Faire**

$c_\theta \leftarrow \arg\min_{k \in \{1,\ldots,K\}} \sum_{\tau=1}^{K} h_{k\tau}(t) F_\theta \sum_{j=1}^{p} \left((\alpha_\theta^j - u_\tau^j(s))^2 + (\beta_\theta^j - v_\tau^j(s))^2\right)$

$C_{c_\theta} \leftarrow \boldsymbol{\mathcal{H}}_\theta$ {□ Attribution du *bin* $\boldsymbol{\mathcal{H}}_\theta$ à la classe C_{c_θ}}

$C_{c_\theta} \leftarrow \boldsymbol{x}_i$ tel que $\xi(i) = \theta$ {□ Attribution de l'observation $\boldsymbol{x}_i$ à la classe C_{c_θ}}

Fin pour

Partition P_s générée

Calcul des valeurs de la fonction de voisinage : h_{kc_θ}, $\theta \in \{1, \ldots, B\}$, $k \in \{1, \ldots, K\}$

Pour $\theta = 1 \rightarrow B$ **Faire**
Pour $k = 1 \rightarrow K$ **Faire**
$h_{kc_\theta}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_\theta} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$
Fin pour
Fin pour
Mise à jour des vecteurs prototypes :
Pour $k = 1 \rightarrow K$ **Faire**
Pour $j = 1 \rightarrow p$ **Faire**
$u_k^j(s+1) \leftarrow \frac{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)\alpha_\theta^j}{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)}$
$v_k^j(s+1) \leftarrow \frac{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)\beta_\theta^j}{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)}$
Fin pour
Fin pour
$s \leftarrow s + 1$
Jusqu'à $G_{binSOM}(\mathbb{W}(s)) == G_{binSOM}(\mathbb{W}(s-1))$ {□ (voir équation (4.1), p.4.1)}
$t \leftarrow t + 1$
Réduire $\sigma(t)$
Jusqu'à $\sigma(t) < \sigma_{final}$
Retourner $\mathbb{W}(f) = \mathbb{W}(s)$
Retourner $P_f = P_s$

La recherche du neurone vainqueur nécessite le plus grand nombre d'opérations arithmétiques de l'algorithme d'apprentissage, ce nombre étant proportionnel à $S.B.K^2.p$ où S est le nombre total de sous-itérations. Donc, la complexité d'un tel algorithme est $O(S.B.K^2.p)$. C'est une complexité linéaire en fonction du nombre de *bins* B mais quadratique en fonction du nombre de neurones K.

4.3.2 Apprentissage heuristique de la carte en mode différé

Il est aussi possible de faire l'apprentissage de la carte en mode différé mais d'une façon heuristique (voir algorithme 4.15, p.152) tout en utilisant le carré de la distance L_2 pour la recherche d'un neurone vainqueur. Dans un tel algorithme, à chaque itération, le rayon de voisinage est réduit et les vecteurs prototypes sont mis à jour. Le nombre total d'itérations doit être fourni à l'avance.

Algorithme 4.15 Apprentissage heuristique en mode différé des cartes auto-organisatrices pour les données discrétisées en se basant sur le carré de la distance L_2 (binSOM-L_2).

Entrées : Fournir :

- $\boldsymbol{\mathcal{R}}_\theta = \left([\alpha_\theta^1, \beta_\theta^1], \ldots, [\alpha_\theta^p, \beta_\theta^p]\right)$, $\theta \in \{1, \ldots, B\}$ {□ Vecteur d'intervalles associé au *bin* $\boldsymbol{\mathcal{H}}_\theta$}
- F_θ {□ Fréquence associée au *bin* $\boldsymbol{\mathcal{H}}_\theta$}
- $\boldsymbol{\xi}$ {□ Vecteur de dimension n, tel que $\xi(i) = \theta$: Appartenance de l'observation $\boldsymbol{x}_i$ à *bin* $\boldsymbol{\mathcal{H}}_\theta$}
- lig, col {□ Dimensions de la carte, $K = lig \cdot col$}
- $\sigma_{init}, \sigma_{final}$ {□ Valeurs initiale et finale du rayon de voisinage}
- T {□ Nombre total d'itérations}

Sorties : Récupérer :

- $\mathbb{W}(T)$ {□ Vecteurs prototypes auto-organisés}
- $P_T = \{C_1, \ldots, C_K\}$ {□ Partition finale}

Initialisation :

$t \leftarrow 0$ {□ t : Itération courante}

Initialiser les vecteurs prototypes $\mathbb{W}(0)$

Apprentissage :

Répéter

$\sigma(t) \leftarrow \sigma_{init} + \left(\frac{t}{T}\right)(\sigma_{final} - \sigma_{init})$

Calcul des neurones vainqueurs des observations :

Pour $\theta = 1 \rightarrow B$ **Faire**

$c_\theta = \arg\min_k \sum_{j=1}^{p} \left((\alpha_\theta^j - u_k^j)^2 + (\beta_\theta^j - v_k^j)^2\right)$

$C_{c_\theta} \leftarrow \boldsymbol{\mathcal{H}}_\theta$ {□ Attribution du *bin* $\boldsymbol{\mathcal{H}}_\theta$ à la classe C_{c_θ}}

$C_{c_\theta} \leftarrow \boldsymbol{x}_i$ tel que $\xi(i) = \theta$ {□ Attribution de l'observation $\boldsymbol{x}_i$ à la classe C_{c_θ}}

Fin pour

Partition P_t générée

Calcul des valeurs de la fonction de voisinage : h_{kc_θ}, $\theta \in \{1, \ldots, B\}$, $k \in \{1, \ldots, K\}$

Pour $\theta = 1 \rightarrow B$ **Faire**

Pour $k = 1 \rightarrow K$ **Faire**

$h_{kc_\theta}(t) \leftarrow \exp\left(-\frac{\|\boldsymbol{r}_{c_\theta} - \boldsymbol{r}_k\|^2}{2\sigma^2(t)}\right)$

Fin pour

Fin pour

Mise à jour des vecteurs prototypes :
Pour $k = 1 \rightarrow K$ **Faire**
Pour $j = 1 \rightarrow p$ **Faire**
$u_k^j(t+1) \leftarrow \frac{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)\alpha_\theta^j}{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)}$
$v_k^j(t+1) \leftarrow \frac{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)\beta_\theta^j}{\sum_{\theta=1}^{B} F_\theta h_{kc_\theta}(t)}$
Fin pour
Fin pour
$t \leftarrow t+1$
Jusqu'à $t > T$
Retourner $\mathbb{W}(T)$
Retourner P_T

La recherche du neurone vainqueur constitue le processus le plus coûteux nécessitant un nombre d'opérations arithmétiques proportionnel à $T.B.K.p$ où T est le nombre total d'itérations. Donc, la complexité d'un tel algorithme est $O(T.B.K.p)$. C'est une complexité linéaire en fonction du nombre de *bins* non vides B et linéaire en fonction du nombre de neurones K. Donc, avec une carte auto-organisatrice pour les données discrétisées, nous gagnons considérablement en temps de calcul du fait que le nombre des *bins* est bien inférieur au nombre des observations. Cependant, le nombre de *bins* doit être choisi de manière à garantir une bonne qualité de classification qui risque de se détériorer si ce nombre est peu élevé (voir section 4.4, p.153).

4.4 Étude expérimentale

Cette étude inclut une série d'expériences conduites sur un jeu de données simulées en vue de valider les méthodes proposées et de les comparer entre elles et à d'autres méthodes. Elle inclut aussi une application sur la segmentation d'images réalisée en proposant une technique de segmentation basée sur les données discrétisées et sur les cartes auto-organisatrices. Il convient de noter que les algorithmes des différentes méthodes sont codés avec Matlab fonctionnant sous le système Windows 7-64 bit avec un processeur Intel Core i5 - 2.3 Ghz et une mémoire RAM de 8 Go.

4.4.1 Jeu de données simulées

Ce jeu de données comprend $n = 100000$ points simulés en utilisant six distributions normales bivariées à proportions égales (chaque classe contient le même nombre d'observations) (voir figure 4.1, p.154). Le tableau 4.1 montre les coordonnées des moyennes de ces distributions ayant toutes la matrice de covariance suivante :

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

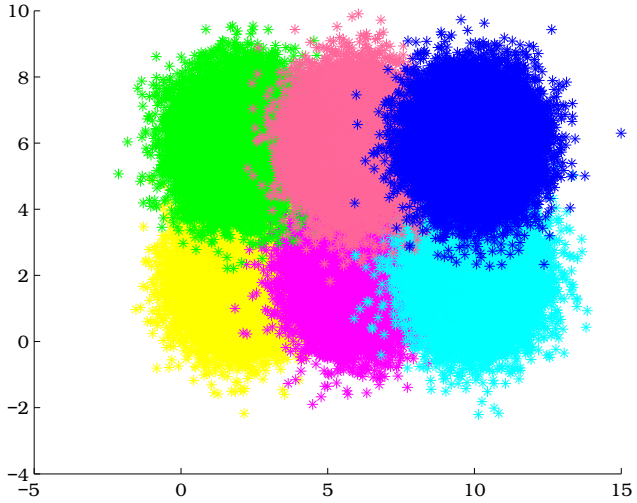


FIGURE 4.1 – Jeu de données simulées de six classes à proportions égales.

Tableau 4.1 – Coordonnées des moyennes des six distributions.

Classe	Moyenne
C1	(2,2)
C2	(6,2)
C3	(10,2)
C4	(2,6)
C5	(6,6)
C6	(10,6)

Ce jeu est regroupé suivant 80 *bins* par dimension pour produire $B = 3674$ *bins* non vides (voir figure 4.2, p.155). Nous proposons de construire une carte auto-organisatrice pour ce jeu de données discrétisées dans le but de les classifier en six

classes. Pour cela, la carte est formée de six neurones arrangés suivant une grille rectangulaire de deux lignes et trois colonnes. L'apprentissage de la carte se fait en premier lieu avec la méthode binSOM-DYN-L_2 (voir algorithme 4.14, p.149) en trois itérations principales : la première correspond à un rayon de voisinage initial égal à $\sigma_{init} = \sigma(1) = 1.5$, la seconde à un rayon de voisinage égal à $\sigma(2) = 0.5$ et la dernière à un rayon de voisinage final égal à $\sigma_{final} = \sigma(3) = 0.1$. Les vecteurs prototypes initiaux sont choisis aléatoirement dans l'ensemble des observations ce qui pourrait influencer les résultats obtenus. Pour cette raison, 20 lancées de cet algorithme sont exécutées avec à chaque fois une initialisation différente pour les vecteurs prototypes, et les résultats produits par la lancée qui permet de minimiser le plus le critère G_{binSOM} (voir équation (4.1), p.148) sont retenus. La figure 4.3 montre les *bins* classifiés en six classes, ainsi que les vecteurs prototypes de la carte qui est bien ordonnée et déployée sur les *bins*.

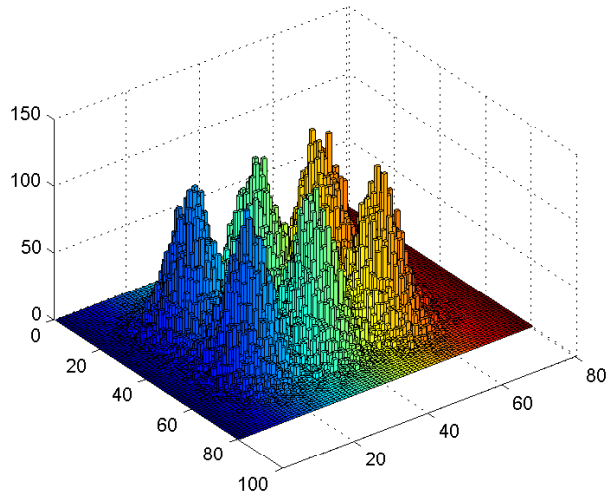


FIGURE 4.2 – Résulats du regroupement en *bins* du jeu de données simulées.

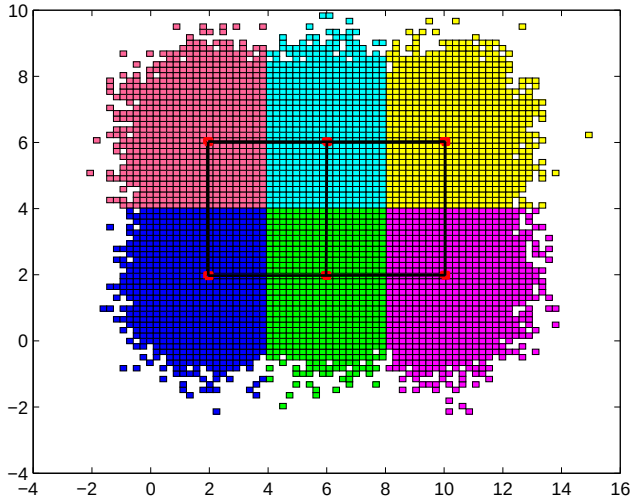


FIGURE 4.3 – Carte auto-organisatrice après apprentissage sur les *bins* classés en 6 classes à proportions égales avec la méthode binSOM-DYN-L_2.

Le même jeu de données est utilisé pour entraîner une carte auto-organisatrice de six neurones, arrangés en deux lignes et trois colonnes, avec la méthode binSOM-L_2 (voir algorithme 4.15, p.152) après avoir regroupé les observations en *bins* en utilisant 80 *bins* par dimensions. Les paramètres d'apprentissage sont : nombre d'itérations total de $T = 100$, rayon de voisinage initial de $\sigma_{init} = 1.5$ et final de $\sigma_{final} = 0.1$. Les vecteurs prototypes initiaux sont choisis aléatoirement en exécutant 20 lancées de l'algorithme avec une initialisation différente à chaque fois, et seulement les résultats de la lancée qui permet de minimiser le plus le critère G_{binSOM} sont pris en compte.

Dans le but de rendre les résultats de classification plus concluants, nous avons simulé 100 réplications de ce jeu de données dans le cadre de simulations de Monte Carlo. Des cartes auto-organisatrices sont construites et entraînées pour ces 100 réplications. Le tableau 4.2 donne la moyenne des $OERC$ (pourcentages des observations mal classifiées) (voir sous-section 1.4.1, p.39), la moyenne des erreurs topographiques te (voir équation (1.24), p.34) et la moyenne du temps d'apprentissage de la carte pour les 100 jeux avec les écarts-types entre parenthèses, et ceci pour les méthodes binSOM-DYN-L_2 et binSOM-L_2, ainsi que pour les méthodes SOM-DYN-L_2 et SOM-L_2 qui sont homologues aux méthodes binSOM-DYN-L_2 et binSOM-L_2 mais qui opèrent sur les observations originales, c'est à dire sur les points au lieu

des *bins*. Nous comparons aussi les méthodes proposées à la méthode des centres-mobiles (où 20 lancées de l'algorithme sont exécutées à chaque fois et seulement les résultats de la lancée qui permet de minimiser le plus le critère des centres-mobiles sont adoptés).

Tableau 4.2 – Résultats obtenus pour le jeu de données simulées avec six classes à proportions égales.

Méthode	***OERC***	***te***	**Temps de classification**
binSOM-DYN-L_2	5.28% (0.07%)	3.1% (12.6%)	8.8s (2s)
binSOM-L_2	5.28% (0.07%)	4.9% (14.6%)	8.2s (0.9s)
SOM-DYN-L_2	5.25% (0.07%)	1.5% (7.8%)	320.4s (43.7s)
SOM-L_2	5.25% (0.07%)	6.8% (15.9%)	250.3s (23.2s)
Centres-mobiles	5.25% (0.07%)		16.3s (2.5s)

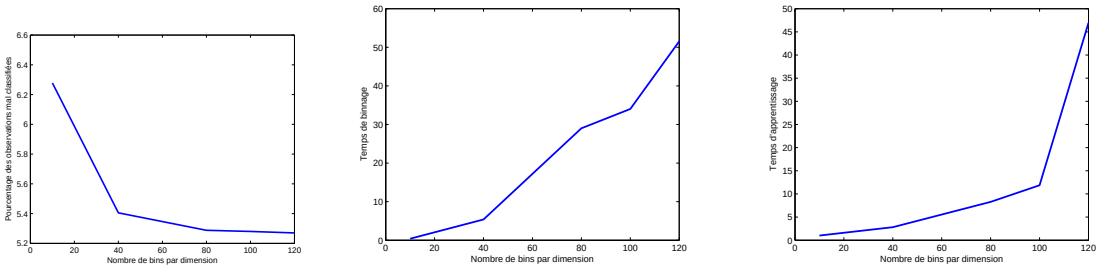
D'après le tableau 4.2 nous remarquons, comme attendu, une réduction considérable du temps d'apprentissage de la carte quand cette dernière est entraînée avec des *bins*, sans pour autant perdre en qualité de classification. En effet, la moyenne des temps d'apprentissage avec la méthode SOM-DYN-L_2 pour les 100 jeux est de 320.4s alors qu'elle est réduite à 6.8s en regroupant les données et en présentant les *bins* à la carte, conduisant ainsi à une moyenne des $OERC$ de 5.28%. Ceci prouve qu'il n'y a pas de perte sensible dans la qualité de la classification. Nous pouvons tirer la même conclusion en comparant les méthodes binSOM-L_2 et SOM-L_2. Nous remarquons aussi que les résultats obtenus avec la méthode des centres-mobiles sont identiques à ceux obtenus avec la méthode SOM-L_2 qui représente l'algorithme standard de Kohonen en mode différé des cartes auto-organisatrices.

Le choix du nombre de *bins* par dimension pourrait influencer les résultats de la classification. Le tableau 4.3 donne les résultats obtenus avec les méthodes binSOM-DYN-L_2 et binSOM-L_2 en faisant varier le nombre de *bins* par dimension.

Tableau 4.3 – Résultats obtenus pour le jeu de données simulées avec six classes à proportions égales en changeant le nombre de *bins* par dimension.

binSOM-DYN-L_2				
Nombre de *bins*	***OERC***	***te***	**Temps de discrétisation**	**Temps d'apprentissage**
10×10	6.68% (4.03%)	6% (19.3%)	0.5s (0.1s)	0.6s (0.1s)
40×40	5.39% (0.12%)	3.3% (11.8%)	6.05s (0.65s)	2.1s (0.3s)
80×80	5.28% (0.07%)	3.1% (12.6%)	27.3s (6.4)	8.8s (2s)
100×100	5.28% (0.07%)	4.6% (14.2%)	37.6s (4.2s)	12.9s (3s)
120×120	5.27% (0.07%)	6.7% (17.5%)	49.7s (1.9s)	17.6s (3.3s)
binSOM-L_2				
Nombre de *bins*	***OERC***	***te***	**Temps de discrétisation**	**Temps d'apprentissage**
10×10	6.27% (0.45%)	0.39% (3.8%)	0.3s (0.01s)	1.0s (0.01s)
40×40	5.40% (0.12%)	5.32% (13.0%)	5.3s (0.4s)	2.8s (0.09s)
80×80	5.28% (0.07%)	4.94% (14.6%)	29.0s (13.9s)	8.2s (0.9s)
100×100	5.28% (0.07%)	7.01% (17.4%)	34.0s (4.9s)	11.8s (1.1s)
120×120	5.27% (0.07%)	8.01% (19.2%)	51.5s (4.8s)	46.9s (6.5s)

Le tableau 4.3 montre qu'en augmentant le nombre de *bins* par dimension, les données seront mieux classifiées mais le temps de discrétisation des données et le temps d'apprentissage de la carte augmentent sensiblement. Toutefois, à partir de 80 *bins* par dimension, le pourcentage des observations mal classifiées décroît lentement alors que les temps d'apprentissage et de binnage croissent très rapidement (voir figures 4.4, p.158). Donc, le nombre de *bins* par dimension doit être choisi judicieusement de façon à éviter des temps longs de binnage et d'apprentissage tout en conservant des résultats de classification acceptables.



(a) *OERC* (b) Temps de discrétisation (c) Temps d'apprentissage

FIGURE 4.4 – Variation du *OERC* et des temps de discrétisation et d'apprentissage en fonction du nombre de *bins* par dimension.

En appliquant les deux méthodes binSOM-DYN-L_2 et bin-SOM-L_2 sur le même jeu de données, mais en changeant les proportions des classes suivant le tableau 4.4 (voir figure 4.5, p.160), **nous remarquons que les six classes ne sont pas identifiées correctement comme le montre la figure 4.6**. Ceci est probablement dû au fait que plus de neurones sont déployés dans les zones de données plus concentrées que d'autres. Pour remédier à ce problème, nous proposons d'utiliser une large carte pour projeter les données en premier lieu, puis classifier les prototypes de la carte en second lieu (voir sous-section 1.3.3, p.36). Nous avons choisi de faire la classification des vecteurs prototypes avec la méthode de la classification hiérarchique ascendante en utilisant le critère de la moyenne défini dans l'équation (4.5) pour déterminer la distance entre deux classes.

$$D(C_1, C_2) = \frac{1}{\big|C_1\big|\big|C_2\big|} \sum_{\boldsymbol{\mathcal{W}}_k \in C_1} \sum_{\boldsymbol{\mathcal{W}}_l \in C_2} d(\boldsymbol{\mathcal{W}}_k, \boldsymbol{\mathcal{W}}_l) \tag{4.5}$$

où $\big|C\big|$ désigne l'effectif de la classe C et d est le carré de la distance L_2 entre deux vecteurs d'intervalles.

Tableau 4.4 – Paramètres des six distributions gaussiennes avec des proportions différentes par classe.

Classe	**Moyenne**	**Taille**
C1	(2,2)	416 points
C2	(6,2)	5000 points
C3	(10,2)	416 points
C4	(2,6)	416 points
C5	(6,6)	3336 points
C6	(10,6)	416 points

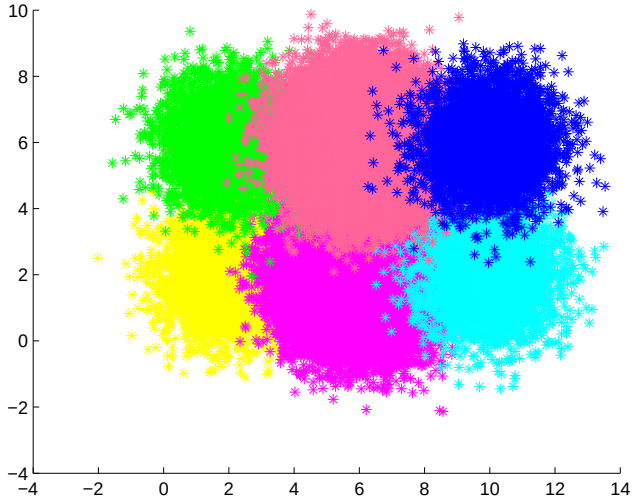


FIGURE 4.5 – Jeu de données simulées de six classes à proportions différentes.

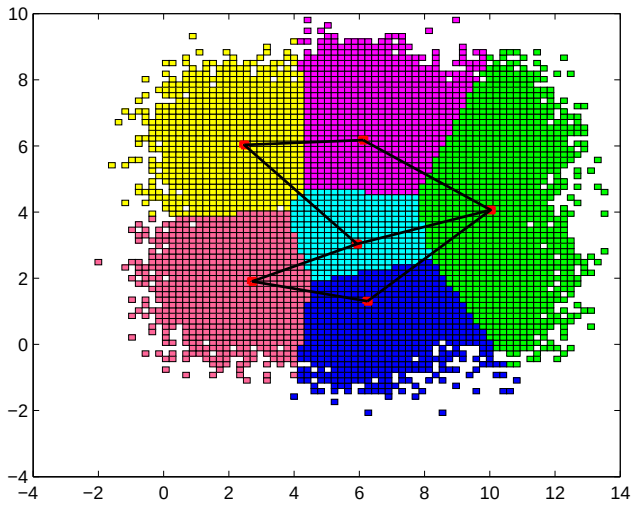


FIGURE 4.6 – Carte auto-organisatrice après apprentissage sur les *bins* classés en 6 classes à proportions différentes.

Les $B = 3574$ *bins* obtenus en regroupant les données avec 80 *bins* par dimension, sont projetés sur une carte auto-organisatrice formée de 294 neurones répartis suivant une grille rectangulaire de 14 lignes et 21 colonnes. L'apprentissage de la carte se fait avec la méthode binSOM-L_2 en $T = 100$ itérations, un

rayon de voisinage initial $\sigma_{init} = 10.5$ et final $\sigma_{final} = 0.1$, ou bien avec la méthode binSOM-DYN-L_2 en 6 itérations correspondantes respectivement aux rayons de voisinage : $\sigma_{init} = \sigma(1) = 10.5$, $\sigma(2) = 7.5$, $\sigma(3) = 4.5$, $\sigma(4) = 1.5$, $\sigma(5) = 0.5$ et $\sigma_{final} = \sigma(6) = 0.1$. Concernant les vecteurs prototypes initiaux, ils sont choisis d'une façon déterministe, par exemple, les premières K observations. Ce choix est dû au fait que la carte auto-organisatrice constitue un résultat intermédiaire avant la classification finale ce qui permet d'atténuer largement le problème de l'optimum local causé par le choix des prototypes initiaux (Dong et Xie, 2005).

La classification finale des données est réalisée en classifiant les prototypes de la carte en 6 classes avec la méthode de la classification hiérarchique ascendante. Chaque observation appartiendra à la classe du prototype du neurone vainqueur du *bin* qui la contient. La figure 4.7 montre le déploiement de la carte sur les *bins* après apprentissage avec la méthode binSOM-L_2+intHC tout en distinguant les six classes résultantes de la classification hiérarchique des vecteurs prototypes. Le tableau 4.5 donne la moyenne des $OERC$ (pourcentage des observations mal classifiées), la moyenne des te (erreurs topographiques), la moyenne des temps de discrétisation et la moyenne des temps d'apprentissage et de classification (avec les écart-types entre parenthèses) en appliquant les deux méthodes binSOM-DYN-L_2+intHC binSOM-L_2+intHC ainsi que la méthode des centres-mobiles sur les 100 réplications du jeu de données simulées avec des distributions gaussiennes à proportions différentes.

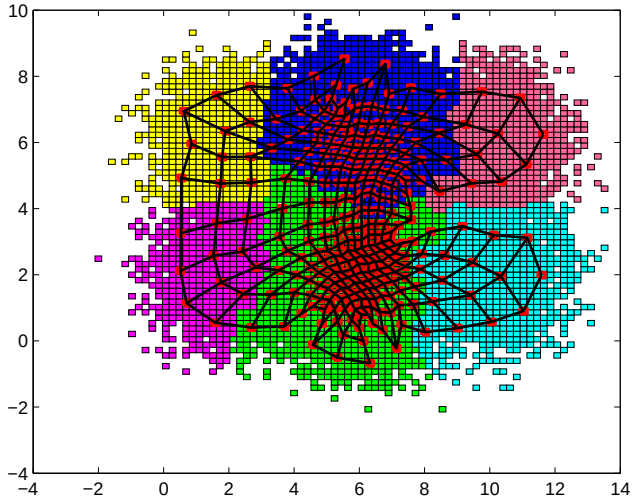


FIGURE 4.7 – Carte auto-organisatrice et classification hiérarchique pour le jeu de données simulées de 6 classes à proportions différentes.

Tableau 4.5 – Résultats obtenus avec une classification en deux étapes pour le jeu de données simulées avec des classes à proportions différentes.

Méthode	***OERC***	***te***	**Temps de discrétisation**	**Temps de classification**
binSOM-DYN-L_2+intHC	7.69% (1.91%)	24.4% (3.3%)	23.2s (2.8s)	64.4s (8.25s)
binSOM-L_2+intHC	7.49% (1.47%)	17.9% (0.6%)	22.7s (0.5s)	28.8s (0.8s)
Centres-mobiles	9.87% (0.25%)			38.3s (12.6s)

Les résultats du tableau 4.5 montrent que les méthodes binSOM-DYN-L_2 et binSOM-L_2 classifient les données plus correctement que la méthode des centres-mobiles en donnant une information supplémentaire sur le voisinage des classes. L'erreur topographique est plus importante dans le cas de la méthode binSOM-DYN-L_2, donc la topologie est mieux préservée avec la méthode binSOM-L_2. De plus, la méthode binSOM-DYN-L_2 est plus coûteuse en temps de classification. Ce qui revient à conclure que la méthode binSOM-L_2 est plus adaptée au cas de cartes auto-organisatrices de grande taille.

4.4.2 Jeux de données réelles - Segmentation d'images

Les cartes auto-organisatrices sont largement utilisées dans la segmentation d'images surtout quand il s'agit d'un processus en deux étapes où une grande carte est construite, dans une première étape, pour la réduction des couleurs, et une classification des vecteurs prototypes de la carte est réalisée dans une deuxième étape pour assurer la classification finale des pixels de l'image (voir sous-section 1.3.3, p.36). Ristić *et al.* (2008), Chi (2011) et Moreira et Costa (1996) ont mis en place une technique de segmentation qui consiste à construire une carte auto-organisatrice puis à utiliser l'algorithme des centres-mobiles pour la classification finale des pixels. Dong et Xie (2005) ont proposé de construire une carte auto-organisatrice puis d'utiliser l'algorithme *Simulated annealing* pour classifier les prototypes de la carte et réaliser la classification finale des pixels. Gonçalves *et al.* (2008) ont développé une technique en deux étapes pour la segmentation des images de télédétection en construisant une carte auto-organisatrice dont les prototypes sont classifiés avec la méthode de la classification hiérarchique agglomérative.

Dans les techniques de segmentation basées sur les cartes auto-organisatrices, l'apprentissage de la carte risque d'être lent, voire impossible, si l'image est de grande taille et le nombre de neurones assez élevé. Une solution à ce problème serait de regrouper les pixels de l'image en *bins* qui seront projetés sur la carte à la place des observations originales.

L'approche que nous proposons dans cette thèse pour segmenter une image est constituée de trois étapes : un regroupement des pixels en *bins* dans une première étape, une carte auto-organisatrice construite pour les *bins* dans une deuxième étape et une classification finale des prototypes de la carte dans une troisième étape. Chaque observation appartiendra à la classe du neurone vainqueur du *bin* qui la contient. Pour l'apprentissage de la carte, nous proposons d'utiliser la méthode binSOM-L_2 (voir algorithme 4.15, p.152) présentant une performance meilleure pour les grandes cartes que la méthode binSOM-DYN-L_2. Pour la classification des vecteurs prototypes, nous proposons d'utiliser la classification hiérarchique ascendante avec le critère de la moyenne (voir équation (4.5), p.159). Dans ce qui suit, nous notons la méthode de segmentation proposée par binSOM-L_2+intHC

Le modèle de couleurs que nous adoptons dans notre approche est le modèle CIE : L^*a^*b où L^* caractérise la couche de luminosité, a^* caractérise la couche de chromaticité indiquant où la couleur se situe le long de l'axe rouge-vert et b^*

caractérise la couche de chromaticité indiquant où la couleur se situe le long de l'axe bleu-jaune. Cet espace colorimétrique est plus approprié pour la séparation des couleurs que le modèle RGB en raison de son homogénéité perceptive. En effet, les relations non linéaires de L^*, a^* et b^* sont destinées à imiter la réponse non linéaire de l'œil de manière à ce que tout changement dans les couleurs soit mieux perçu visuellement qu'avec le modèle RGB.

Puisque toute l'information sur les couleurs est contenue dans l'espace a^*b^*, les individus à classifier sont alors des pixels avec les valeurs a^* et b^* correspondantes. Pour tester la méthode proposée, nous avons choisi deux images. La première est celle d'un arbre faisant partie de la bibliothèque Berkeley pour la segmentation d'images accessible sur :
$http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/$
La deuxième est celle d'une chaise faisant partie des *101 object categories* accessible en ligne à travers l'adresse :
$http://www.vision.caltech.edu/Image\_Datasets/Caltech101/Caltech101.html$

Segmentation de l'image Arbre

L'image Arbre (voir figure 4.8, p.165) est constituée de 321×481 pixels que nous allons classifier suivant leurs couleurs. Après conversion de l'image de l'espace RGB vers l'espace $L^*a^*b^*$, les observations ayant comme abscisse a^* et comme ordonnée b^* sont discrétisées en *bins* à raison de 100 *bins* par dimension pour produire $B = 2714$ *bins* non vides. Cette étape de regroupement des données constitue une première réduction des couleurs. Ces *bins* sont projetés sur une carte auto-organisatrice carrée de 100 neurones pour une deuxième réduction des couleurs. L'apprentissage de la carte se fait en $T = 100$ itérations avec des valeurs initiale et finale du rayon de voisinage de $\sigma_{init} = 5$ et $\sigma_{final} = 0.1$. Les vecteurs prototypes initiaux sont égaux aux premières K observations. La figure 4.9 montre la carte auto-organisatrice obtenue après apprentissage. La couleur de chaque neurone est obtenue en calculant la moyenne des couleurs de tous les pixels appartenant aux *bins* ayant le neurone en question comme neurone vainqueur. Le premier neurone de la carte (neurone 1) est le neurone situé le plus en haut et à gauche alors que son opposé sur la carte est le dernier neurone (neurone 100).

FIGURE 4.8 – Image Arbre.

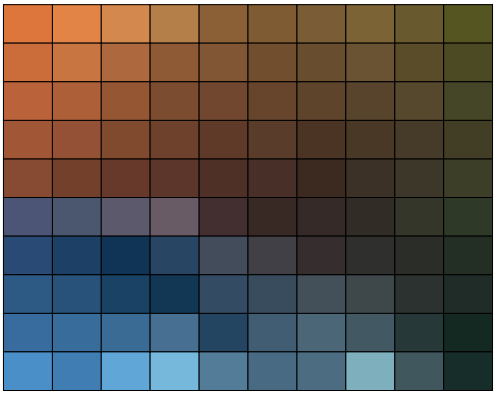


FIGURE 4.9 – Carte auto-organisatrice après apprentissage pour projeter les *bins* de l'image Arbre.

La figure 4.10 représente le dendrogramme résultant de la classification hiérarchique ascendante des vecteurs prototypes de la carte. Pour éviter l'encombrement du dendrogramme, nous avons choisi d'afficher sur l'axe horizontal 30 nœuds seulement en affichant dans le tableau 4.6 les neurones associés à chaque nœud. En coupant le dendrogramme de façon à avoir 2 classes, les pixels de l'image seront classifiés suivant deux couleurs. La figure 4.11 représente le résultat de la segmentation

de l'image Arbre avec la méthode binSOM-L_2+intHC en affichant les pixels de la première classe en noir et les pixels de la deuxième classe en blanc. Nous remarquons une séparation parfaite entre les pixels représentant le ciel à couleur bleue et le reste des pixels de l'image. Ce n'est pas le cas en segmentant la même image avec l'algorithme des centres-mobiles qui est lancé 20 fois avec à chaque fois une initialisation différente afin d'éviter le problème de l'optimum local (voir figure 4.12, p.168).

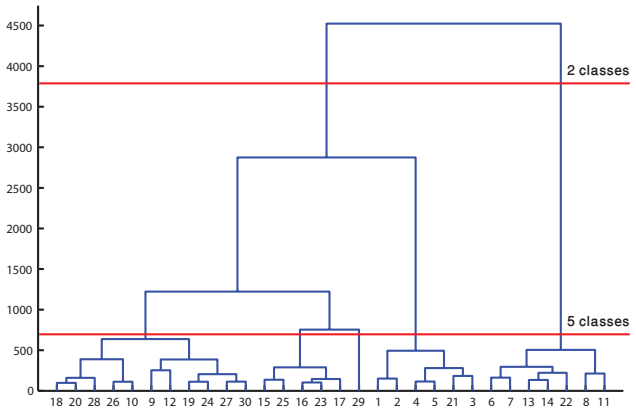


FIGURE 4.10 – Dendrogramme des prototypes de la carte pour l'image Arbre.

Tableau 4.6 – Neurones associés à chacun des 30 nœuds du dendrogramme de l'image Arbre.

Nœud	**Neurones**	**Nœud**	**Neurones**
1	1 11	16	51 61 71
2	2 3 12	17	52 53 62 63 72 73
3	31	18	57 78 88
4	4 13 22	19	74 84
5	5 14 23	20	76 77 87
6	6 16 48	21	21
7	7 17 18 27 28 37 38 39 49	22	80
8	8 9 10 19 20 29 30	23	81
9	35 45 54 55 64	24	82 83
10	36	25	25 34 43 44
11	40	26	26
12	46 56 65 66 67 75	27	85 86 96
13	47 58 68 69 79	28	89 90 97 98 99 100
14	50 59 60 70	29	91 92
15	15 24 32 33 41 42	30	93 94 95

FIGURE 4.11 – Segmentation en 2 couleurs de l'image Arbre avec la méthode binSOM-L_2+intHC.

FIGURE 4.12 – Segmentation en 2 couleurs de l'image Arbre avec la méthode des centres-mobiles.

Si l'on désire connaître le nombre de couleurs ou de classes optimal pour segmenter l'image Arbre, nous effectuons plusieurs classifications des prototypes de la carte en faisant varier le nombre de classes à chaque fois et en évaluant le résultat de chaque classification obtenue en calculant un critère interne (voir sous-section 1.4.2, p.41). Nous proposons d'utiliser une version modifiée du critère Davies-Bouldin pour évaluer le résultat de la classification pour des vecteurs d'intervalles. Le nombre de classes conduisant à une valeur minimale de ce critère est supposé être le plus convenable. L'expression du critère Davies-Bouldin étendu aux intervalles est donné par :

$$intDB = \frac{1}{M}\sum_{m=1}^{M} R_m \qquad (4.6)$$
$$R_m = \max_{\substack{m' \in \{1,\ldots,M\} \\ m' \neq m}} R_{mm'}$$

où M est le nombre total de classes et $R_{mm'}$ est une mesure de similarité entre les deux classes C_m et $C_{m'}$ calculée selon :

$$R_{mm'} = \frac{s_m + s_{m'}}{D(C_m, C_{m'})}$$

où :

– $D(C_m, C_{m'})$ est la mesure de dissimilarité entre les deux classes C_m et $C_{m'}$ déterminée en calculant la distance entre les centres de gravité des deux classes.

Si $\boldsymbol{\varpi}_m = \left([\nu_m^1, \vartheta_m^1], \ldots, [\nu_m^p, \vartheta_m^p]\right)$ et $\boldsymbol{\varpi}_{m'} = \left([\nu_{m'}^1, \vartheta_{m'}^1], \ldots, [\nu_{m'}^p, \vartheta_{m'}^p]\right)$ sont les centres de gravité des deux classes C_m et $C_{m'}$ respectivement, cette mesure de dissimilarité est calculée en utilisant le carré de la distance L_2 pour les données intervalles :

$$\begin{aligned} D(C_m, C_{m'}) &= d(\boldsymbol{\varpi}_m, \boldsymbol{\varpi}_{m'}) \\ D(C_m, C_{m'}) &= \sum_{j=1}^{p} \left((\nu_m^j - \nu_{m'}^j)^2 + (\vartheta_m^j - \vartheta_{m'}^j)^2\right) \end{aligned}$$

– s_m est la mesure de dispersion de la classe C_m calculée par :

$$\begin{aligned} s_m &= \frac{\sum_{k=1}^{|C_m|} d(\boldsymbol{\mathcal{W}}_k, \boldsymbol{\varpi}_m)}{|C_m|} \\ s_m &= \frac{\sum_{k=1}^{|C_m|} \sum_{j=1}^{p} \left((u_k^j - \nu_m^j)^2 + (v_k^j - \vartheta_m^j)^2\right)}{|C_m|} \end{aligned}$$

où $|C_m|$ est le cardinal de la classe C_m.

La figure 4.13 donne l'indice $intDB$ calculé sur le résultat de la classification hiérarchique ascendante appliquée aux 100 prototypes de la carte en fonction du nombre de classes. Un nombre de classes égal à 5 permet d'obtenir la plus petite valeur pour l'indice $intDB$.

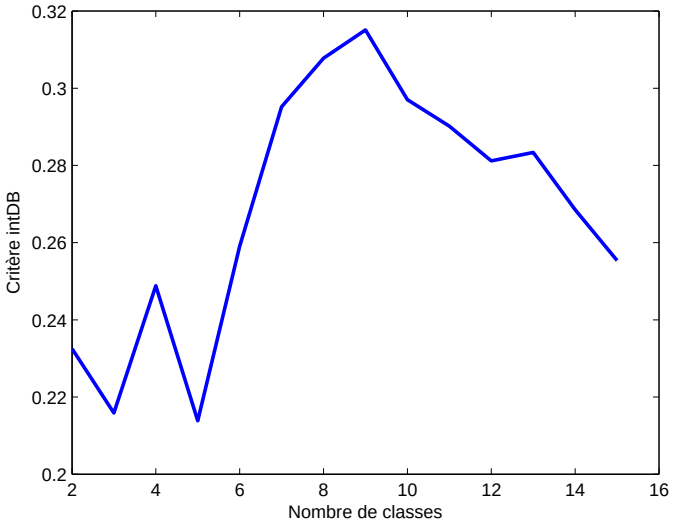


FIGURE 4.13 – Variation de l'indice $intDB$ en fonction du nombre de classes pour l'image Arbre.

Les figures 4.14 et 4.15 représentent le résultat de la segmentation de l'image Arbre avec la méthode binSOM-L_2+intHC et la méthode des centres-mobiles (20 lancées) respectivement. Il est clair que le résultat obtenu avec la méthode binSOM-L_2 est meilleur que celui obtenu avec la méthode des centres-mobiles. La méthode proposée (binSOM-L_2+intHC) permet de mieux reproduire l'image Arbre avec 5 couleurs.

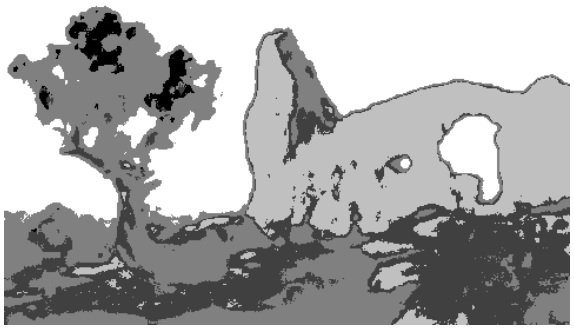


FIGURE 4.14 – Segmentation en 5 couleurs de l'image Arbre avec la méthode binSOM-L_2+intHC.

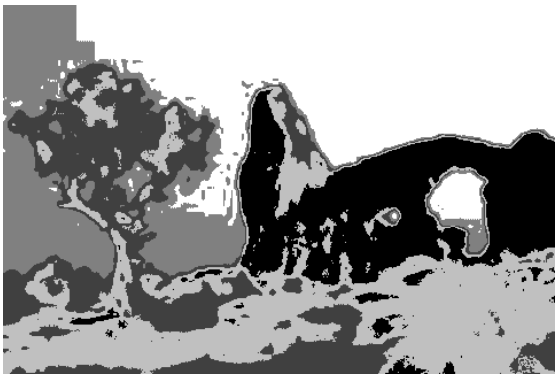


FIGURE 4.15 – Segmentation en 5 couleurs de l'image Arbre avec la méthode des centres-mobiles.

Le tableau 4.7 donne les temps d'exécution des méthodes binSOM-L_2+intHC,

de la méthode SOM+HC et de la méthode des centres-mobiles pour segmenter l'image Arbre, en faisant varier le nombre de couleurs entre 2 et 15. La méthode SOM+HC représente une projection des pixels sur une carte carrée de 100 neurones suivie d'une classification hiérarchique classique (appliquée aux points) avec le critère de la moyenne qui est appliqué aux prototypes de la carte.

Tableau 4.7 – Temps d'exécution pour segmenter l'image Arbre.

Méthode	**binSOM-L_2+intHC**	**SOM+HC**	**Centres-mobiles(20 rép.)**
Discrétisation des données	45.33s		
Apprentissage de la carte	5.84s	336.960	
14 classifications (de 2 à 15 classes)	0.39s	0.095s	354.22s
Temps total	51.56s	337.05s	354.22s

D'après les résultats obtenus, nous pouvons conclure que la méthode proposée (binSOM-L_2+intHC) permet de donner des résultats satisfaisants quant à la segmentation de l'image Arbre en un temps assez limité en comparaison avec d'autres méthodes de segmentation.

Segmentation de l'image Chaise

L'image Chaise (voir figure 4.16, p.172) est constituée de 300×273 pixels. Les observations ayant comme abscisse a^* et comme ordonnée b^* sont discrétisées à raison de 80 *bins* par dimension pour produire $B = 3106$ *bins* non vides (voir figure 4.17, p.172). Ces *bins* sont projetés sur une carte auto-organisatrice carrée de 100 neurones. L'apprentissage de la carte se fait suivant la méthode binSOM-L_2 (voir figure 4.18, p.173) avec les paramètres suivants : $T = 100$ itérations, $\sigma_{init} = 5$ et $\sigma_{final} = 0.1$. Les vecteurs prototypes initiaux sont égaux aux premières K observations.

FIGURE 4.16 – Image Chaise.

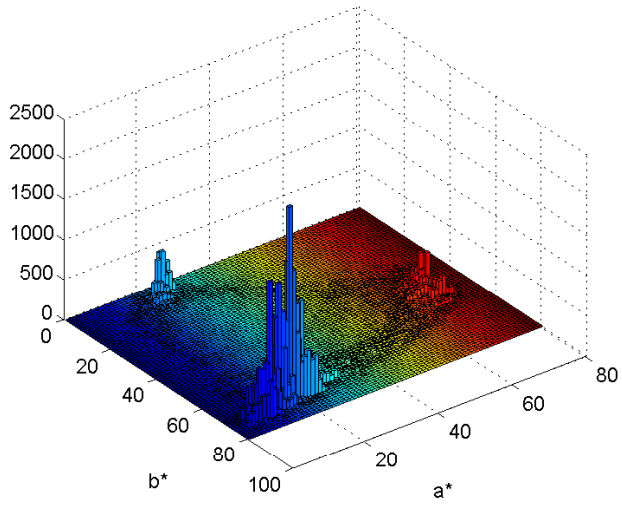


FIGURE 4.17 – Regroupement en *bins* de l'image Chaise.

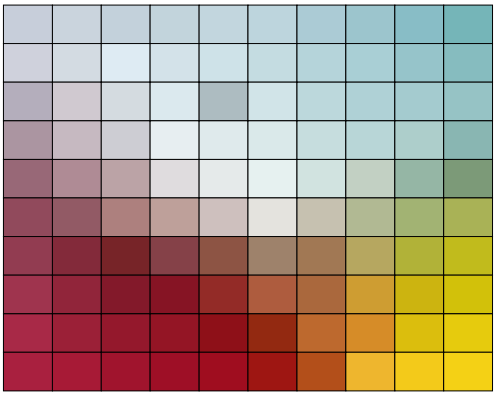


FIGURE 4.18 – Carte auto-organisatrice après apprentissage pour projeter les *bins* de l'image Chaise.

La méthode de la classification hiérarchique ascendante est appliquée aux 100 prototypes de la carte. Pour les 14 partitions obtenues, correspondant à un nombre de classes allant de 2 à 15, le critère interne $intDB$ (voir équation (4.6), p.168) est calculé. Il atteint une valeur minimale avec 3 classes (voir figure 4.19, p.174). Donc, une segmentation de l'image Chaise avec 3 couleurs semble être la plus convenable. D'ailleurs, la figure 4.17 montre le groupement des *bins* en 3 classes. En effet, ces 3 classes correspondent aux 3 couleurs : rouge, bleu et jaune. Les figures 4.20 et 4.21 représentent le résultat de la segmentation de l'image chaise avec la méthodes binSOM-L_2+intHC et la méthode des centres-mobiles (20 lancées) respectivement. Les deux résultats sont très comparables avec trois couleurs et les deux méthodes ont pu classifier correctement les pixels de l'image en trois classes. Par contre, avec une segmentation en 8 couleurs, la méthode binSOM-L_2+intHC (voir figure 4.22, p.175) permet une meilleure séparation des couleurs qu'avec la méthode des centres-mobiles (voir figure 4.23, p.176).

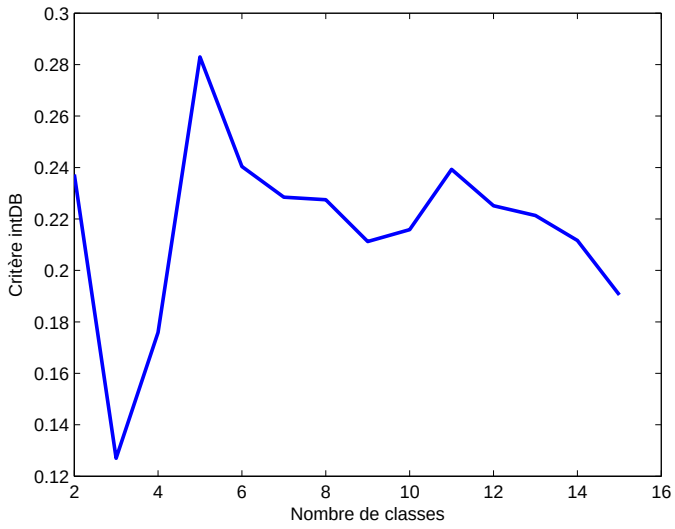


FIGURE 4.19 – Variation de l'indice $intDB$ en fonction du nombre de classes pour l'image Chaise.

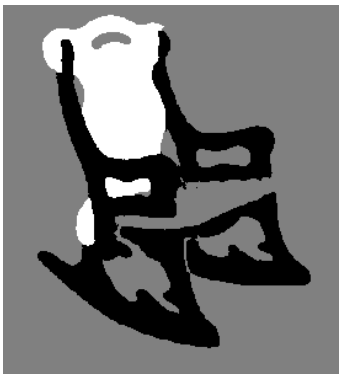


FIGURE 4.20 – Segmentation en 3 couleurs de l'image Chaise avec la méthode binSOM-L_2+intHC.

FIGURE 4.21 – Segmentation en 3 couleurs de l'image Arbre avec la méthode des centres-mobiles.

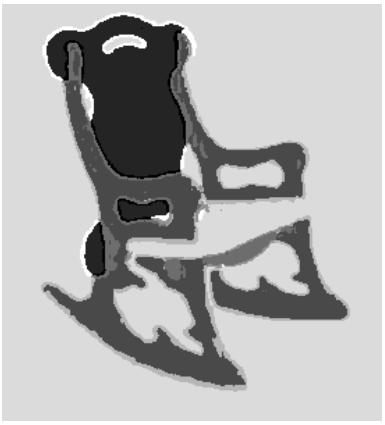


FIGURE 4.22 – Segmentation en 8 couleurs de l'image Chaise avec la méthode binSOM-L_2+intHC.

FIGURE 4.23 – Segmentation en 8 couleurs de l'image Arbre avec la méthode des centres-mobiles.

Les résultats obtenus pour les deux images Arbre et Chaise ont prouvé les capacités de segmentation de la méthode binSOM-L_2+intHC. Précisons que la méthode de segmentation proposée donne toujours les mêmes résultats à chaque exécution du fait que les vecteurs prototypes initiaux de la carte sont choisis de manière déterministe.

4.5 Conclusion

Dans ce chapitre, nous avons abordé le problème de la classification de données discrétisées. Classifier les *bins* à la place des observations originales permet de diminuer la taille de l'échantillon d'entrée et de réduire la complexité algorithmique du programme de classification. Les méthodes de classification pour les données discrétisées, que nous avons proposées dans ce chapitre, se fondent sur les cartes auto-organisatrices. Dans les problèmes de classification où le nombre de classes est connu *a priori* et où les classes ne diffèrent pas trop en taille, une petite carte où le nombre de neurones est le même que le nombre de classes pourrait être utilisée. Cependant, dans la plupart des problèmes de classification non supervisée, nous ne disposons d'aucune information concernant le nombre de classes, ni leurs proportions. Pour ces raisons, il vaut mieux construire une grande carte pour projeter les

bins tout d'abord, et faire ensuite classifier les prototypes de la carte, pour réaliser la classification finale des observations. Nous avons appliqué cette technique dans la segmentation en couleurs d'images où les pixels de l'image à segmenter sont regroupés en *bins*, projetés sur grande carte, puis une classification hiérarchique ascendante est appliquée aux prototypes de la carte pour une classification finale des pixels. Les résultats obtenus sont assez encourageants, surtout que le temps d'exécution de l'algorithme correspondant est assez limité favorisant son utilisation pour des images de grande taille. Pourtant, pour les données à haute dimensionnalité ($p > 5$), le regroupement des données en *bins* devient coûteux en temps, voire irréalisable, ce qui pourrait constituer une limitation majeure de cette méthode.

Conclusion générale

Le travail mené dans cette thèse est axé sur la classification des données symboliques avec des méthodes automatiques à approche géométrique, plus spécifiquement avec les cartes auto-organisatrices.

Récapitulatif

Après avoir donné, dans le **premier chapitre**, un aperçu général sur les méthodes usuelles de la classification automatique, nous avons détaillé plus longuement les méthodes à approche géométrique. Dans ce contexte, nous avons évoqué les principes de la classification hiérarchique ascendante et la méthode des centres-mobiles et ses variantes. Nous nous sommes intéressés plus particulièrement aux cartes auto-organisatrices vu leur rôle important dans la classification des données, tout en donnant une information supplémentaire sur la proximité des classes en préservant la topologie des données. Plusieurs algorithmes proposés dans le cadre de cette thèse se fondent sur les cartes auto-organisatrices. Nous avons inclus dans ce premier chapitre une explication détaillée du principe des cartes, de leurs algorithmes d'apprentissage et de leur rôle dans la classification des données quantitatives univaluées.

Le **deuxième chapitre** a constitué une première migration du formalisme classique des données vers leur formalisme symbolique. Après avoir défini les différents types de données symboliques et les mesures de proximité qui permettent de les comparer, nous avons cité les différentes méthodes de partitionnement existantes dans la littérature pour ces types de données. Nous avons proposé deux approches, basées sur les cartes auto-organisatrices, pour classifier des données décrites par des variables symboliques mixtes. Dans la première approche, l'apprentissage de la carte est précédé d'une phase d'homogénéisation des données afin de les transformer en

histogrammes qui sont utilisés ensuite pour entraîner la carte. Dans la deuxième approche, l'apprentissage de la carte se fait conformément à l'algorithme des cartes auto-organisatrices pour les dissimilarités, en présentant à la carte des distances calculées en utilisant une distance adéquate entre observations symboliques mixtes. Les deux méthodes ont prouvé leur validité en les testant et en les comparant à d'autres méthodes. Il est cependant nécessaire de noter que l'homogénéisation risque de causer des distorsions dans les données dans la première méthode, et la construction de la matrice de distances dans la deuxième méthode nécessite un espace mémoire important qui croît fortement avec la taille de l'échantillon.

En raison de l'utilisation croissante des données de type intervalle, qui sont une sorte de données symboliques, essentiellement utilisées pour modéliser l'incertitude et la variabilité dans les données, nous avons axé le **troisième chapitre** sur la classification de données intervalles. Après avoir exposé les différentes méthodes existantes pour classifier les données intervalles, nous avons mis en place plusieurs algorithmes d'apprentissage en mode différé des cartes auto-organisatrices pour classifier des données intervalles. Ces algorithmes peuvent se scinder en deux groupes. Dans le premier groupe, l'apprentissage est conduit en optimisant un certain critère, alors que dans le deuxième groupe, l'apprentissage se fait d'une façon heuristique à la manière de l'algorithme standard des cartes auto-organisatrices. Quand l'apprentissage de la carte se fait en optimisation un critère, l'utilisateur ne doit plus fournir le nombre d'itérations à l'algorithme, ce qui peut constituer un avantage majeur par rapport aux algorithmes du second groupe. Cependant, la complexité algorithmique des méthodes du premier groupe est plus importante. La validité des méthodes proposées a été illustrée en les comparant à d'autres méthodes de classification de données intervalles, moyennant des jeux de données simulées et réelles, dont deux ont été créés dans le cadre de cette thèse.

Quand le nombre d'observations est assez élevé, l'apprentissage d'une carte auto-organisatrice pour les données ponctuelles devient lent. Pour cette raison, nous avons proposé d'utiliser, dans le **quatrième chapitre**, le formalisme des données discrétisées *(binned data)* qui sont une sorte de données symboliques résultantes du regroupement des observations en histogramme multidimensionnel. Nous avons alors mis en place une carte auto-organisatrice pour ces types de données tout en présentant deux algorithmes d'apprentissage opérant sur les données discrétisées. De même, nous avons proposé une technique de classification des données ponctuelles en trois étapes : regroupement des données en première étape, projection des *bins*

sur la carte auto-organisatrice en deuxième étape et classification des prototypes de la carte en troisième étape. Les avantages de cette technique ont été illustrés en la testant sur un jeu de données simulées et en l'appliquant à la segmentation d'images.

Avantages et limites

Les travaux proposés dans cette thèse présentent plusieurs avantages, entre autres :

- Les algorithmes proposés ont une complexité algorithmique réduite permettant leur utilisation sur des grands jeux de données.
- Le formalisme des données symboliques permet de mieux modéliser les informations issues d'expériences réelles.
- La préservation de la topologie permet de renseigner sur la proximité des classes : deux neurones voisins sur la carte auto-organisatrice représentent des observations voisines dans l'espace d'entrée.
- L'utilisation de distances de Mahalanobis adaptatives pour construire une carte auto-organisatrice pour les données intervalles conduit à une classification adaptative des observations. En plus, la distance de Mahalanobis prend en compte la variabilité des variables et la corrélation entre les variables, et elle permet la reconnaissance de classes elliptiques (voir sous-section 3.6.3, p.111).
- Le regroupement des données en *bins* (voir section 4.2, p.147) permet de gagner en temps de calcul et rend possible la classification des très grands jeux de données.
- La segmentation d'images en trois étapes (voir sous-section 4.4.2, p.163) permet de gagner en temps de calcul, sans perte de qualité, rendant ainsi possible la segmentation des images de très grande taille.

Cependant, quelques contraintes et limites sont à considérer, à savoir :

- Le nombre de classes doit être fourni à l'avance.
- Les classes produites, suite à l'utilisation des cartes auto-organisatrices, sont plutôt convexes. Par exemple, la recherche de classes concentriques ne peut être réalisée.
- Quand la recherche du neurone vainqueur pour des données intervalles est

basée sur la distance de Mahalanobis, des problèmes d'inversion de la matrice de covariance peuvent mener à une partition inadéquate.
- Le regroupement des données à haute dimensionalité (p>5) conduit à un grand nombre de *bins* nécessitant un espace mémoire important et rendant l'algorithme de regroupement complexe.

Perspectives

Dans ce qui suit, nous proposons quelques idées pour compléter et perfectionner les travaux effectués dans cette thèse :

- Proposer un algorithme d'apprentissage dynamique des cartes auto-organisatrices où la réduction du rayon de voisinage se réalise dans le cadre d'un processus qui s'adapte aux données traitées.
- Proposer un mécanisme à deux étapes pour la classification des données symboliques (voir chapitre 2) et des données intervalles (voir chapitre 3) à la lumière des avantages de cette technique introduite dans le chapitre 4, qui consiste à projeter les données sur la carte auto-organisatrice et à classifier ensuite les prototypes de la carte pour achever la classification finale des données.
- Élargir les expérimentations à d'autres jeux de données en testant les méthodes proposées afin de prouver leur robustesse.
- Rendre possible le regroupement des données pour les jeux de données à haute dimensionalité en le précédant d'une Analyse en Composantes Principales dans le but de faire une réduction de dimensions.
- Développer d'autres algorithmes de classification automatique pour les données discrétisées comme la classification hiérarchique par exemple.
- Étendre d'autres méthodes de classification automatique pour prendre en compte les données symboliques, comme à titre d'exemple, la classification à base de densité et la classification à base de graphes. Ces méthodes permettent de détecter automatiquement le nombre de classes et de produire des classes de formes quelconques (voir section 1.2, p.8). L'extension des ces méthodes aux données symboliques serait possible en utilisant une distance adéquate entre données symboliques dans le but de construire la matrice de distances requise par ces algorithmes. L'extension du *Support Vector Cluste-*

ring (voir sous-section 1.2.3, p.11) aux données intervalles est également possible en remplaçant la distance euclidienne, utilisée dans la fonction noyau, par une distance entre vecteurs d'intervalles (Do et Poulet, 2009).

Annexe A

Jeux de données de type intervalle

Cet annexe contient les deux jeux de données intervalles concernant les températures minimales et maximales collectées en France et au Liban pour l'année 2010.

Tableau A.1 – Jeu de données France-météo.

Obs.	Villes	Jan.	Fév.	Mar.	Avr.	Mai	Juin	Juil.	Août	Sep.	Oct.	Nov.	Déc.
1	Abbeville	[-1.5,3]	[0.9,5.9]	[3.6,10]	[4.5,15]	[5.8,15]	[11,21]	[14,24]	[14,21]	[11,19]	[7.7,15]	[4.2,8.8]	[-2.4,2.2]
2	Agen	[0.1,7.1]	[1.4,10]	[3.2,14]	[6.2,20]	[9,20]	[13,25]	[17,29]	[15,28]	[11,24]	[7.9,19]	[5.6,13]	[-0.6,7.8]
3	Ajaccio	[5.4,12]	[5.4,14]	[5.8,16]	[8.4,19]	[12,21]	[15,25]	[19,30]	[17,28]	[15,26]	[12,22]	[8.4,17]	[5.5,14]
4	Albi	[-0.1,7.1]	[0.6,9.8]	[3.3,14]	[6.4,20]	[9.4,20]	[14,25]	[17,30]	[15,29]	[12,25]	[8.3,18]	[5,12]	[-1.3,8.6]
5	Alençon	[-2,3.5]	[0.9,7]	[2.4,11]	[3.7,16]	[6.6,17]	[11,23]	[14,26]	[12,23]	[8.8,21]	[7.1,16]	[3.7,9.2]	[-1.6,3]
6	Ambérieu	[-2.7,2.7]	[-0.3,7.2]	[1.3,12]	[5.3,19]	[8.6,19]	[13,24]	[16,30]	[14,26]	[8.7,22]	[7.3,16]	[4.5,11]	[-2.2,5]
7	Angers	[-0.1,5.4]	[1.7,8.7]	[3.8,13]	[5.8,18]	[8.3,19]	[13,24]	[15,28]	[14,25]	[11,23]	[8.2,17]	[5.3,11]	[-0.7,4.9]
8	Aubenas	[-0.9,5.7]	[0.4,8.6]	[2.9,13]	[7.1,19]	[9.9,20]	[14,26]	[18,31]	[16,29]	[12,24]	[8.3,17]	[4.5,12]	[0.4,6.9]
9	Auch	[0.1,7.6]	[0.4,9.9]	[2.9,14]	[5.4,20]	[9.1,20]	[13,24]	[17,29]	[15,27]	[11,24]	[8.2,19]	[5.3,13]	[-0.9,9]
10	Aurillac	[-3,3.1]	[-1.6,5.7]	[0.4,10]	[4.2,16]	[6.2,16]	[10,21]	[14,26]	[11,24]	[8.1,20]	[5.8,15]	[2.7,8]	[-2.8,5.6]
11	Auxerre	[-1.9,2.7]	[0.7,6.9]	[2.7,12]	[5.7,17]	[8.4,18]	[13,24]	[16,28]	[14,24]	[10,21]	[6.7,16]	[4.9,9.3]	[-1.5,2.5]
12	Bâle-Mulhouse	[-3.1,1.7]	[-0.9,5.8]	[1.4,11]	[4.2,18]	[8.6,17]	[12,24]	[15,29]	[14,24]	[8,20]	[5,15]	[3,9.3]	[-3.3,2.5]
13	Bastia	[4.6,12]	[4.8,14]	[6.1,15]	[8.6,19]	[12,21]	[16,26]	[21,30]	[19,29]	[16,26]	[12,21]	[8.1,17]	[4.8,13]
14	Beauvais	[-2.2,2.9]	[0.3,6.1]	[2.5,11]	[3.1,16]	[5.5,17]	[11,23]	[14,26]	[12,23]	[9,20]	[6.4,15]	[3.4,8.8]	[-3.7,1.7]
15	Belfort	[-3.5,0.4]	[-1,4.7]	[1.4,9.9]	[4.7,17]	[8.2,16]	[12,23]	[15,27]	[13,22]	[9,19]	[5.8,14]	[3.2,7.9]	[-3.4,1.7]
16	Belle-Île	[3.5,7.6]	[4.5,8.7]	[5.3,10]	[8.5,15]	[10,16]	[14,21]	[16,21]	[15,20]	[13,20]	[12,16]	[8.3,12]	[3,7.7]
17	Bergerac	[-0.9,6.5]	[0.7,9.8]	[2.5,14]	[5.5,20]	[8.7,20]	[13,25]	[15,29]	[13,27]	[9.3,24]	[6.7,19]	[4.9,12]	[-0.8,8.1]
18	Besançon	[-3,2]	[0,6.1]	[1.7,11]	[5.6,17]	[8.6,17]	[13,23]	[16,27]	[13,23]	[9.4,20]	[6.2,15]	[4.1,9]	[-1.7,3.6]
19	Biarritz	[4,10]	[4.3,11]	[5.8,15]	[8.7,18]	[11,18]	[14,21]	[18,24]	[16,25]	[14,23]	[10,19]	[8,14]	[2.6,11]
20	Biscarrosse	[2.5,8.9]	[4,11]	[5.5,14]	[9.6,19]	[12,19]	[15,22]	[18,26]	[16,25]	[14,23]	[11,19]	[8.5,14]	[2.8,9.9]
21	Blois	[-2.1,3.4]	[0.7,7.1]	[2.5,12]	[4.7,17]	[7,18]	[12,24]	[14,28]	[13,25]	[9.9,21]	[6.9,16]	[3.9,9.4]	[-1.2,3.1]
22	Bordeaux	[0.8,6.9]	[2.4,10]	[4.3,14]	[8.3,20]	[10,20]	[14,25]	[17,28]	[15,27]	[12,24]	[9.2,19]	[6.8,13]	[1.4,8.3]
23	Boulogne-sur-Mer	[-0.1,3.6]	[1.8,6]	[4.1,9.1]	[6.5,13]	[7.7,13]	[12,19]	[15,21]	[15,19]	[13,18]	[9.8,15]	[5.4,8.7]	[-0.5,3.3]
24	Bourg-Saint-Maurice	[-5.4,2.9]	[-3.7,5.9]	[0.2,12]	[3.8,18]	[7.2,18]	[12,24]	[14,28]	[12,24]	[8.1,21]	[5.6,16]	[0.8,9.3]	[-4.3,3.2]
25	Bourges	[-1.7,3.5]	[0.7,7.4]	[2.4,12]	[6.3,18]	[8.3,18]	[13,24]	[15,27]	[14,25]	[11,21]	[6.8,17]	[4.9,9.8]	[-1.1,4.1]
26	Brest-Guipavas	[1.3,7.1]	[2.5,8.7]	[3.8,11]	[5.1,15]	[8.1,16]	[11,20]	[14,21]	[13,20]	[11,19]	[9.4,16]	[5.6,11]	[0.8,7.1]
27	Brive-la-Gaillarde	[-0.8,6.1]	[0.1,9.7]	[1.8,14]	[4.8,20]	[8.1,19]	[12,25]	[16,29]	[13,27]	[9.1,23]	[6.5,18]	[5,11]	[-1.4,8.3]
28	Cap-de-la-Hève	[0.5,4.5]	[2.5,6.7]	[4.9,9.8]	[6.9,14]	[8.8,15]	[13,19]	[16,22]	[15,19]	[13,19]	[10,15]	[5.9,9.4]	[0.7,4.4]
29	Carcassonne	[1,7.4]	[1.8,9.9]	[4.1,13]	[7.7,20]	[10,20]	[15,25]	[18,30]	[16,29]	[13,24]	[10,18]	[5.8,13]	[1.8,9]
30	Cazaux	[0.7,8.3]	[2,11]	[3.2,15]	[7,20]	[9.7,20]	[14,24]	[16,27]	[14,26]	[11,24]	[8.2,19]	[6.6,14]	[1.2,9.3]
31	Chambéry	[-2.3,3.4]	[0.3,7.2]	[1.8,12]	[5.1,19]	[9.6,19]	[13,24]	[16,30]	[14,26]	[9.2,22]	[6.9,16]	[3.1,11]	[-2.6,4.1]
32	Charleville-Mézières	[-3.1,1.6]	[-0.8,5.2]	[0.9,11]	[1.4,16]	[5,17]	[9.6,23]	[12,27]	[11,22]	[7.7,19]	[4.9,15]	[3.4,8.5]	[-3.9,1.1]
33	Chartres	[-2.3,2.9]	[0.6,6.8]	[2.3,12]	[3.9,17]	[6.7,18]	[12,23]	[14,27]	[13,24]	[9.1,22]	[6.5,16]	[3.8,9.1]	[-1.9,2.1]
34	Châteauroux	[-2.1,3.7]	[0.3,7.6]	[1.8,13]	[4.6,18]	[7,18]	[12,23]	[14,27]	[13,25]	[9.1,22]	[6.3,17]	[4.6,9.8]	[-1.8,4]
35	Cherbourg-Valognes	[0.4,5.6]	[2.4,8]	[3.8,11]	[4.7,14]	[6.7,16]	[10,20]	[13,23]	[13,20]	[10,19]	[8.6,15]	[4.3,10]	[-0.1,5.3]
36	Clermont-Ferrand	[-2.3,4]	[0.2,7.5]	[2.1,12]	[5.2,18]	[7.7,17]	[12,22]	[15,27]	[14,25]	[9.4,21]	[7,16]	[4.5,11]	[-2,5.5]
37	Cognac	[-0.2,6.1]	[2.2,9.5]	[3.9,14]	[7.6,19]	[9.7,20]	[14,25]	[16,28]	[14,26]	[11,24]	[8.7,19]	[6.1,12]	[0.5,7.3]
38	Colmar	[-4,1.3]	[-0.7,6]	[1.8,12]	[4.5,18]	[8.8,18]	[13,24]	[15,29]	[14,24]	[8.4,21]	[4.6,15]	[3.7,9.7]	[-3.4,2.1]
39	Dax	[2.5,9.4]	[3.1,11]	[4,16]	[7.9,21]	[10,20]	[14,24]	[17,28]	[15,28]	[13,25]	[8.5,19]	[7,14]	[1.6,10]
40	Dijon	[-3,2.2]	[0,6.3]	[1.6,11]	[5.1,17]	[8.5,17]	[13,23]	[15,28]	[14,24]	[9.4,20]	[6.1,16]	[4.3,9.3]	[-2.6,3.1]
41	Dinard	[1.2,6.2]	[2.4,8.5]	[4.1,12]	[5.1,15]	[7.8,16]	[12,20]	[14,23]	[14,21]	[11,20]	[9.6,16]	[5.4,11]	[0.6,5.9]
42	Dunkerque	[0.3,3.6]	[1.5,5.9]	[4.7,10]	[6.8,13]	[8.8,14]	[13,19]	[16,23]	[16,21]	[13,19]	[10,15]	[5.9,9.6]	[-0.1,3.5]
43	Embrun	[-5.3,3.4]	[-3.8,5.5]	[-0.1,11]	[3.8,17]	[6.3,18]	[11,24]	[15,30]	[12,27]	[8.2,22]	[5.3,17]	[0.7,9.6]	[-4.7,3.6]
44	Epinal	[-3.7,1.1]	[-0.2,5.2]	[1.1,10]	[2.6,17]	[7.3,16]	[11,23]	[14,27]	[13,22]	[8.1,19]	[4.4,14]	[4,8.2]	[-3.1,2]
45	Evreux	[-2,3.2]	[0.6,6.7]	[2.9,11]	[3.8,16]	[6.3,17]	[11,23]	[14,27]	[13,23]	[9.8,21]	[7.1,15]	[3.7,9.1]	[-2.7,2.1]
46	Gourdon	[-0.8,5.4]	[0.7,8.8]	[2.9,14]	[6.4,19]	[8.3,19]	[13,24]	[15,28]	[13,27]	[9.8,23]	[7.6,17]	[4.6,11]	[-0.3,7.8]
47	Grenoble	[-3.7,2.5]	[-0.1,7.5]	[2.4,13]	[6,20]	[9.6,19]	[14,25]	[17,30]	[14,26]	[10,22]	[7.5,17]	[3.2,11]	[-3.3,4]
48	Guéret	[-2.2,2.1]	[-0.5,5]	[2.1,10]	[6.6,16]	[7.8,15]	[12,21]	[16,25]	[14,23]	[11,21]	[7.5,15]	[3.7,7.7]	[-1.7,3.5]
49	Îled'Ouessant	[4.7,8.3]	[5.1,9.4]	[6.4,10]	[7.8,13]	[9.7,15]	[13,18]	[14,20]	[14,19]	[13,18]	[12,16]	[8,12]	[5,8.3]
50	Îled'Yeu	[2.1,7.2]	[3.5,9]	[4.8,11]	[8.2,17]	[10,17]	[14,22]	[16,23]	[15,22]	[12,21]	[11,17]	[7.8,13]	[2.3,7.5]
51	LaRoche-sur-Yon	[-0.6,5.8]	[1.3,8.7]	[3.3,12]	[6.1,18]	[8.1,18]	[13,24]	[14,26]	[13,24]	[10,22]	[8.4,16]	[5.1,11]	[-0.5,5.3]
52	LaRochelle	[0.9,6.4]	[3.1,8.6]	[5.1,12]	[8.8,17]	[11,18]	[15,23]	[18,24]	[16,23]	[13,21]	[10,17]	[7,12]	[1.6,6.6]
53	Langres	[-3.7,0.5]	[-1,4]	[1.2,9]	[5.2,15]	[7.5,15]	[12,22]	[15,26]	[13,22]	[9.4,18]	[5.7,14]	[3.3,6.9]	[-3.4,1.2]
54	LeLuc	[0.5,9.6]	[2.8,12]	[4.7,16]	[6.6,21]	[11,23]	[15,28]	[18,34]	[18,32]	[12,27]	[9.3,20]	[5.9,15]	[2,10]
55	LeMans	[-1.1,4.7]	[1.3,8.3]	[3.3,13]	[5.4,18]	[8.6,19]	[13,24]	[15,28]	[14,25]	[10,22]	[7.7,17]	[4.4,10]	[-0.7,4.4]
56	LePuy	[-4.9,1]	[-3.6,3.9]	[-1.4,8.1]	[2.4,15]	[4.8,14]	[9.4,20]	[12,25]	[10,24]	[6.1,19]	[4.5,13]	[1.5,7.8]	[-4.8,3.7]
57	LeTouquet	[-0.9,4.1]	[1.1,6.9]	[3.2,11]	[4.6,15]	[5.7,15]	[11,20]	[14,22]	[14,20]	[11,19]	[8.3,15]	[4.6,9.4]	[-1.7,3.7]
58	Lille	[-1.9,2.5]	[0.1,5.4]	[3.2,11]	[4.8,16]	[6.6,16]	[12,22]	[15,26]	[13,22]	[11,19]	[7.5,15]	[4.1,8.6]	[-2.5,1.4]
59	Limoges	[-1.7,3.2]	[0.2,6.1]	[2.6,11]	[7.1,16]	[8.5,16]	[13,21]	[16,25]	[13,23]	[11,20]	[7.8,15]	[4.5,8.4]	[-0.9,4.6]

Obs.	Villes	Jan.	Fév.	Mar.	Avr.	Mai	Juin	Juil.	Août	Sep.	Oct.	Nov.	Déc.
60	Lons-le-Saunier	[-2.6,2]	[0.5,6.4]	[2.7,11]	[6.4,18]	[9,17]	[13,23]	[17,27]	[15,24]	[10,20]	[7.4,16]	[4.9,9.2]	[-1.4,4.5]
61	Lorient	[0.5,7]	[1.8,8.8]	[3.5,11]	[6.1,16]	[8.1,17]	[12,22]	[14,23]	[13,22]	[9.8,20]	[9.3,17]	[5.3,12]	[0.2,6.8]
62	Luxeuil	[-3.7,1.7]	[-0.3,6.1]	[0.7,11]	[2.6,18]	[7.7,17]	[12,24]	[14,28]	[13,23]	[8,20]	[4.5,16]	[3.3,8.6]	[-2.8,2.7]
63	Lyon-Bron	[-1.2,3.3]	[1.2,7.6]	[3.3,12]	[7.7,19]	[10,19]	[15,25]	[18,30]	[16,27]	[11,22]	[8.4,17]	[5.7,11]	[-0.3,5.9]
64	Mâcon	[-1.6,2.7]	[0.6,6.6]	[2.6,12]	[6.9,18]	[9.8,18]	[14,24]	[16,28]	[15,25]	[9.7,21]	[7.2,16]	[4.9,10]	[-1.2,4.3]
65	Marignane	[1.7,8.5]	[3.2,11]	[5.6,15]	[9.2,20]	[13,22]	[17,27]	[21,32]	[19,30]	[14,25]	[11,20]	[7,15]	[2.9,10]
66	Melun	[-2,2.9]	[0.9,6.9]	[3,12]	[4.9,17]	[7.9,18]	[12,24]	[15,27]	[14,23]	[10,20]	[7.1,16]	[4.8,9.4]	[-1.9,2.2]
67	Mende	[-5,1.3]	[-4.3,3]	[-2.2,7.1]	[2.1,14]	[4.4,13]	[8.6,19]	[12,25]	[10,23]	[6.6,19]	[3.7,12]	[0.1,6.4]	[-4.4,3.6]
68	Metz	[-2.7,1.2]	[-0.1,5.9]	[2.3,11]	[4.1,17]	[7.7,17]	[12,25]	[15,28]	[13,23]	[9.5,19]	[5.6,15]	[4.6,8.8]	[-3.2,0.8]
69	Millau	[-2.1,3]	[-1.1,5.3]	[1.2,9.3]	[5.6,17]	[7.3,16]	[12,21]	[16,28]	[14,26]	[10,21]	[7.1,14]	[3.2,8.1]	[-1.2,5.2]
70	Mont-Aigoual	[-6.8,-2.7]	[-6.1,-1.9]	[-3.5,0.7]	[1.9,7.4]	[2.7,8.2]	[7.4,14]	[12,19]	[9.8,17]	[6.7,13]	[2.3,7.2]	[-1.5,2.5]	[-5.8,0]
71	Mont-de-Marsan	[0.4,8.2]	[1,11]	[2.4,15]	[5.7,20]	[8.6,20]	[13,25]	[16,29]	[13,28]	[11,25]	[7,19]	[5.7,13]	[-0.6,10]
72	Montauban	[0.1,7.4]	[1.5,10]	[3.5,14]	[6.6,20]	[10,20]	[14,25]	[17,29]	[16,28]	[12,24]	[7.9,19]	[5.1,12]	[-1.2,8.1]
73	Montélimar	[0.6,4.9]	[1.7,8.9]	[4.4,13]	[8.2,19]	[11,20]	[15,26]	[19,31]	[17,29]	[13,24]	[9.8,18]	[6.1,13]	[1.6,6.5]
74	Montpellier	[1.4,8.6]	[2.7,11]	[5,14]	[8.8,19]	[13,22]	[16,26]	[21,31]	[19,30]	[14,25]	[11,19]	[6.9,15]	[2.6,10]
75	Nancy-Essey	[-2.8,1.5]	[0,5.8]	[2.2,11]	[3.9,17]	[7.8,17]	[13,24]	[15,28]	[14,23]	[9,19]	[5.1,15]	[4.5,8.7]	[-3.2,1]
76	Nantes	[-0.3,6.3]	[1.6,9.2]	[3.6,12]	[6.6,18]	[8.9,19]	[13,24]	[14,27]	[13,24]	[10,23]	[8.5,17]	[4.8,11]	[-0.7,5.8]
77	Nevers	[-2.4,3.3]	[0,7.2]	[1.1,13]	[3.5,18]	[6.3,18]	[11,23]	[14,27]	[12,24]	[7.7,21]	[5.2,16]	[4.5,9.9]	[-2.1,4.2]
78	Nice	[4.5,11]	[5.2,12]	[8,14]	[11,18]	[14,20]	[18,24]	[23,29]	[20,27]	[17,24]	[13,20]	[8.9,15]	[5.3,11]
79	Nîmes-Courbessac	[0.6,7.7]	[2.1,11]	[5,14]	[8.6,20]	[12,22]	[16,27]	[20,32]	[18,31]	[14,26]	[10,19]	[5.7,14]	[2.3,9.4]
80	Niort	[-0.9,5.4]	[1.2,8.7]	[3.2,13]	[6.3,18]	[8.5,19]	[13,24]	[15,27]	[13,26]	[11,23]	[8.1,18]	[5.4,11]	[-0.3,5.5]
81	Orange	[-0.6,6.2]	[1.4,10]	[3.7,13]	[7.5,20]	[11,21]	[15,28]	[19,32]	[17,30]	[13,25]	[9.5,19]	[5.3,14]	[1.5,8]
82	Orléans-Bricy	[-2.4,3.1]	[0.4,6.9]	[2.3,12]	[4.4,17]	[7,18]	[12,23]	[14,27]	[13,24]	[9.5,21]	[6.6,16]	[3.5,9.2]	[-1.5,2.7]
83	Paris-Montsouris	[0,3.8]	[2.2,7.3]	[4.8,12]	[7.5,18]	[9.9,18]	[15,24]	[18,27]	[15,24]	[12,21]	[9,16]	[5.7,9.7]	[-0.2,3.1]
84	Pau	[2.1,8.6]	[2.2,11]	[4.1,14]	[7.7,19]	[9.7,19]	[14,23]	[16,26]	[15,26]	[12,23]	[7.6,19]	[5.7,13]	[0.3,10]
85	Perpignan	[3,10]	[2.7,11]	[5.8,14]	[9.6,20]	[13,21]	[17,27]	[21,31]	[20,30]	[16,25]	[12,20]	[7.2,15]	[2.9,11]
86	Poitiers	[-1.6,4.3]	[0.7,7.9]	[2.7,12]	[5.3,18]	[7.8,18]	[12,23]	[15,27]	[13,25]	[9.7,22]	[7.2,17]	[4.8,10]	[-1.1,4.1]
87	Reims	[-2.5,2]	[0.2,6.3]	[2,11]	[3.5,16]	[6.6,17]	[11,23]	[14,27]	[13,23]	[9.2,20]	[5.8,16]	[4.6,8.8]	[-3.5,1.5]
88	Rennes	[-0.4,6]	[1.4,9.2]	[3.3,12]	[4.8,18]	[7.5,18]	[12,23]	[14,27]	[13,23]	[9.8,22]	[8.3,17]	[4.5,11]	[-0.7,5.2]
89	Romorantin	[-2.5,4.3]	[0,8.1]	[1.4,13]	[3.6,19]	[6.2,19]	[11,24]	[13,28]	[12,25]	[7,22]	[5.2,17]	[3.8,9.9]	[-1.4,4.1]
90	Rouen	[-1.8,3.3]	[0.7,6.5]	[3,11]	[4.2,16]	[6.4,17]	[11,22]	[14,26]	[12,22]	[9.6,20]	[7.3,15]	[3.5,8.6]	[-2.3,2.1]
91	Saint-Auban	[-1.9,4.9]	[-0.6,8.2]	[2.4,13]	[5.4,19]	[9,20]	[13,26]	[17,31]	[16,29]	[11,24]	[7.5,18]	[3.7,12]	[-0.5,6.6]
92	Saint-Brieuc	[1.1,6.2]	[2.6,8.2]	[4.2,11]	[5.2,14]	[7.6,15]	[11,19]	[14,23]	[13,21]	[11,20]	[9.7,16]	[5.5,11]	[1.1,5.7]
93	Saint-Dizier	[-2.1,2.5]	[0.9,6.6]	[2.6,12]	[4.3,17]	[7.9,18]	[12,24]	[15,28]	[14,23]	[9.8,20]	[6,16]	[4.9,8.7]	[-2.1,2.2]
94	Saint-étienne	[-2.4,3.1]	[0.1,6.8]	[1.5,11]	[5.1,17]	[7.8,17]	[13,23]	[16,28]	[14,25]	[9.4,21]	[6.8,16]	[4.5,11]	[-1.8,5.5]
95	Saint-Girons	[-0.3,7.2]	[-0.1,9.3]	[2.4,13]	[5.4,18]	[7.8,18]	[12,23]	[15,26]	[14,25]	[10,23]	[7.1,18]	[2.9,12]	[-1.3,9.7]
96	Saint-Quentin	[-2,1.7]	[0.3,5.4]	[2.8,11]	[4,15]	[6.4,16]	[11,22]	[14,26]	[12,22]	[9.9,19]	[6.5,15]	[3.8,8.4]	[-3.5,0.8]
97	Saint-Raphaël	[2.5,10]	[4.1,12]	[6.3,15]	[8.3,19]	[12,22]	[16,26]	[20,31]	[18,28]	[14,25]	[11,20]	[7.3,16]	[3.7,11]
98	Salon-de-Provence	[0.2,8]	[1.5,11]	[3.5,14]	[6.6,20]	[10,21]	[14,27]	[18,32]	[17,30]	[12,25]	[9.5,19]	[5,14]	[1.5,9.3]
99	Solenzara	[5.6,12]	[6.2,15]	[7.4,15]	[9.7,19]	[13,22]	[17,26]	[21,31]	[20,29]	[17,26]	[13,21]	[9,17]	[6.7,13]
100	Strasbourg	[-2.6,1]	[-0.4,5.9]	[2.1,11]	[4.8,18]	[9.1,17]	[14,25]	[16,28]	[14,24]	[9.3,20]	[5,15]	[4.6,9.5]	[-3.1,1.1]
101	Tarbes	[0.4,8.1]	[0.2,9.5]	[2.4,13]	[6.1,18]	[8.2,18]	[13,22]	[16,26]	[15,25]	[12,22]	[7.2,18]	[4.4,13]	[-0.6,9.3]
102	Toulon	[4.1,10]	[5.6,12]	[7.4,15]	[10,20]	[13,22]	[17,26]	[21,32]	[20,29]	[16,26]	[13,20]	[8.9,15]	[5.6,11]
103	Toulouse-Blagnac	[0.8,7.3]	[1.9,9.8]	[4.3,13]	[7.6,19]	[10,20]	[15,25]	[18,29]	[16,28]	[13,24]	[9.8,18]	[5.7,13]	[0.8,8.6]
104	Tours	[-1.2,4.2]	[1.2,7.5]	[3.3,12]	[6.1,17]	[8.4,18]	[13,23]	[15,27]	[14,25]	[11,22]	[7.7,16]	[5,9.8]	[-0.3,3.8]
105	Troyes	[-2.4,2.8]	[0,6.7]	[1.6,12]	[2.9,17]	[6.8,18]	[11,23]	[13,28]	[13,24]	[9,20]	[5.4,16]	[4.2,9]	[-2.7,2.1]
106	Vichy	[-2.4,3.8]	[-0.2,7.6]	[1,13]	[3.4,18]	[7.1,18]	[11,23]	[15,28]	[13,25]	[8.3,21]	[6,17]	[4.3,11]	[-3.1,5.1]

Tableau A.2 – Jeu de données Liban-météo.

Obs.	Villes	Jan.	Fév.	Mar.	Avr.	Mai	Juin	Juil.	Août	Sep.	Oct.	Nov.	Déc.
1	TelAmara	[2.3,13.5]	[2.3,14.2]	[3.7,19.1]	[5.7,22.1]	[9.2,27.0]	[11.9,29.9]	[13.2,34.1]	[15.8,36.2]	[12.3,31.4]	[10.0,26.6]	[4.0,24.0]	[0.9,15.9]
2	Fanar	[11.6,19.2]	[11.3,19.3]	[12.9,21.9]	[14.3,23.0]	[17.3,25.9]	[20.5,28.5]	[22.6,30.2]	[24.7,32.2]	[23.1,30.9]	[20.4,28.7]	[17.5,26.0]	[12.9,21.2]
3	Ghazir	[11.6,19.2]	[11.3,19.3]	[12.9,21.9]	[14.3,23.0]	[17.3,25.9]	[20.5,28.5]	[22.6,30.2]	[24.7,32.2]	[23.1,30.9]	[20.4,28.7]	[17.5,26.0]	[12.9,21.2]
4	HawchAmmik	[3.0,14.3]	[3.1,14.8]	[4.7,19.2]	[6.2,22.2]	[9.3,27.1]	[12.9,29.8]	[14.4,33.8]	[16.4,36.3]	[13.5,31.2]	[10.9,26.2]	[3.8,24.7]	[3.1,19.4]
5	Kaa	[5.6,15.2]	[4.3,14.8]	[7.2,20.4]	[8.6,24.2]	[11.4,28.8]	[15.1,32.0]	[15.7,34.3]	[18.1,36.2]	[16.5,32.2]	[12.7,28.0]	[5.5,23.4]	[3.9,15.6]
6	Kfarchakhna	[9.7,18.4]	[9.0,18.4]	[10.4,21.5]	[12.0,23.1]	[15.0,26.7]	[18.6,29.6]	[20.1,30.9]	[22.7,33.3]	[20.7,30.8]	[18.1,28.6]	[14.6,26.6]	[10.3,20.6]
7	Lebaa	[9.6,18.2]	[9.4,18.8]	[10.6,21.2]	[11.6,22.3]	[14.5,25.2]	[17.9,28.1]	[19.6,28.5]	[21.6,31.0]	[19.5,29.4]	[17.5,28.0]	[14.1,26.4]	[10.2,20.9]
8	Nabatyeh	[9.8,18.2]	[9.3,18.6]	[10.8,21.1]	[11.5,22.8]	[14.9,27.2]	[18.2,30.0]	[20.3,30.7]	[22.3,33.2]	[19.9,31.2]	[17.7,29.7]	[14.7,27.8]	[13.6,21.0]
9	Shheem	[9.5,16.3]	[9.1,17.0]	[11.1,20.0]	[12.1,21.9]	[15.4,25.3]	[17.9,27.8]	[18.8,28.5]	[21.5,31.3]	[19.6,29.0]	[17.9,27.2]	[16.1,25.7]	[11.2,19.2]
10	Rachaya F.	[7.4,13.9]	[7.3,14.5]	[9.7,18.8]	[11.0,21.8]	[13.7,25.6]	[19.0,27.4]	[18.4,30.0]	[21.7,32.6]	[18.5,30.2]	[16.8,27.3]	[14.8,25.0]	[8.8,17.0]
11	Rmeich	[7.8,16.5]	[7.1,17.1]	[8.5,21.0]	[9.1,23.9]	[12.6,27.7]	[15.8,30.5]	[17.9,31.8]	[20.2,34.6]	[17.3,31.8]	[15.2,29.4]	[13.0,26.4]	[8.9,18.7]
12	Saida	[12.5,19.7]	[12.4,20.2]	[13.8,22.2]	[15.0,22.7]	[18.3,25.2]	[21.7,28.3]	[24.0,29.8]	[25.7,31.9]	[24.1,30.7]	[20.8,28.6]	[17.2,26.0]	[13.6,21.9]
13	Tyr	[12.5,19.7]	[12.4,20.2]	[11.1,21.6]	[12.4,22.4]	[15.7,25.3]	[19.2,28.0]	[21.9,29.9]	[23.9,31.8]	[21.6,30.5]	[18.3,28.3]	[13.0,26.2]	[10.4,22.2]
14	SirElDonieh	[8.4,13.9]	[7.3,13.8]	[9.8,17.2]	[11.5,18.9]	[15.4,22.0]	[17.7,24.6]	[19.0,26.1]	[22.3,29.2]	[19.5,25.5]	[16.6,23.0]	[16.1,21.7]	[10.2,15.9]
15	Talia	[2.7,12.1]	[2.7,12.6]	[4.5,17.1]	[5.8,20.6]	[10.1,25.9]	[13.0,28.6]	[14.3,33.1]	[16.8,35.1]	[12.9,30.1]	[10.7,24.9]	[6.8,22.1]	[3.2,14.1]
16	Tarchich	[3.0,8.8]	[1.8,8.4]	[4.8,12.7]	[7.0,15.2]	[10.3,19.7]	[13.4,22.8]	[16.8,26.0]	[20.1,28.5]	[14.5,23.5]	[11.7,20.2]	[9.5,18.0]	[4.5,10.9]
17	Watahoub	[5.4,11.9]	[4.6,11.5]	[7.7,16.0]	[9.2,18.3]	[12.8,22.9]	[15.5,25.3]	[18.6,28.2]	[21.1,30.2]	[16.5,26.4]	[13.8,23.2]	[12.1,21.1]	[7.2,13.9]
18	Aamatour	[8.6,15.7]	[8.3,16.0]	[10.1,19.0]	[11.5,21.5]	[14.8,25.6]	[17.3,27.9]	[18.3,29.1]	[21.4,32.0]	[18.8,29.2]	[17.3,26.9]	[15.4,25.1]	[10.2,18.3]
19	Aarsal	[3.7,11.2]	[3.7,10.9]	[7.0,15.7]	[8.5,18.5]	[13.2,23.7]	[16.5,26.9]	[19.6,30.9]	[21.9,32.0]	[16.7,27.8]	[13.6,22.6]	[9.6,19.5]	[4.3,12.9]
20	AinElAbou	[5.6,12.7]	[5.2,12.5]	[7.8,16.2]	[9.3,18.8]	[12.5,23.1]	[15.1,25.3]	[17.8,27.8]	[20.4,30.8]	[16.8,27.1]	[14.6,24.3]	[13.3,22.1]	[7.7,15.1]
21	Akoura	[4.0,11.0]	[3.8,11.1]	[6.2,14.5]	[7.9,16.5]	[11.2,21.2]	[14.0,23.9]	[17.0,26.7]	[20.1,29.4]	[15.3,25.5]	[12.8,21.9]	[11.0,19.8]	[6.0,13.3]
22	Baakline	[7.8,13.9]	[7.6,14.1]	[9.6,17.1]	[11.0,19.0]	[14.6,22.6]	[16.9,25.0]	[18.1,26.1]	[21.5,29.1]	[18.2,25.9]	[16.6,24.0]	[15.0,22.5]	[9.6,16.2]
23	Btedii	[3.3,12.9]	[3.9,13.4]	[6.6,18.7]	[8.2,21.8]	[12.3,26.9]	[15.3,29.9]	[17.6,34.8]	[20.2,36.1]	[16.1,31.6]	[12.9,26.2]	[9.5,23.3]	[4.3,15.6]
24	ByrHassan	[14.0,20.0]	[13.7,19.9]	[15.3,22.4]	[16.9,23.2]	[19.6,25.5]	[22.7,28.2]	[25.1,29.7]	[27.2,32.1]	[25.9,31.1]	[22.9,29.4]	[20.1,27.1]	[15.2,22.3]
25	Derdgheiya	[10.8,17.6]	[10.1,17.8]	[11.9,20.7]	[13.1,22.4]	[15.7,25.6]	[19.0,28.4]	[20.7,29.1]	[23.0,31.8]	[21.2,30.0]	[19.1,28.6]	[17.1,26.9]	[12.5,20.2]
26	Ehden	[4.4,10.4]	[3.7,9.8]	[6.9,13.8]	[8.2,15.4]	[11.5,19.5]	[14.7,22.2]	[17.5,25.3]	[20.0,27.4]	[14.9,22.6]	[12.8,19.9]	[10.7,17.9]	[5.6,12.3]
27	Fnaydek	[4.8,12.0]	[4.1,11.9]	[6.7,15.2]	[8.2,17.1]	[11.5,21.0]	[14.5,23.3]	[17.2,26.3]	[19.8,29.2]	[15.4,24.4]	[13.0,21.4]	[10.8,20.4]	[5.9,14.6]
28	Hermil	[5.1,12.9]	[5.1,13.5]	[8.0,19.2]	[10.1,23.3]	[13.6,28.6]	[17.3,31.8]	[18.4,35.0]	[21.1,36.7]	[18.2,32.0]	[15.5,27.2]	[10.1,22.0]	[5.8,14.6]
29	H.ElOmara	[2.5,14.1]	[2.9,14.7]	[4.5,19.4]	[6.0,22.4]	[8.7,26.8]	[11.2,28.8]	[12.4,32.2]	[14.3,34.5]	[11.4,30.5]	[9.1,26.4]	[1.9,24.4]	[-0.2,16.4]
30	BarElias	[2.1,14.3]	[2.2,15.0]	[3.8,19.6]	[5.1,22.6]	[8.0,26.9]	[11.2,28.8]	[12.4,32.2]	[14.3,34.5]	[11.4,30.5]	[9.1,26.4]	[1.9,24.4]	[-0.2,16.4]
31	Jabouleh	[3.8,13.0]	[3.9,14.4]	[6.6,19.4]	[8.1,22.7]	[11.8,28.0]	[15.8,31.5]	[17.0,34.1]	[18.2,35.6]	[15.7,31.6]	[12.9,26.8]	[7.6,22.6]	[3.5,13.8]
32	Kherbit A.	[4.4,13.3]	[4.4,13.8]	[6.8,18.0]	[8.5,21.0]	[12.0,25.8]	[15.3,28.0]	[17.4,31.6]	[19.3,34.3]	[15.4,30.4]	[12.9,25.7]	[8.4,24.3]	[4.4,16.5]
33	Machghara	[4.8,13.4]	[4.9,13.9]	[7.3,18.0]	[8.9,20.9]	[11.9,24.8]	[15.5,27.4]	[16.5,31.1]	[18.7,33.6]	[14.9,29.6]	[12.8,25.7]	[8.0,23.8]	[4.4,16.0]
34	Markaba	[8.5,14.1]	[8.1,14.9]	[10.5,18.3]	[11.8,21.3]	[14.4,24.5]	[17.3,27.2]	[18.5,27.9]	[21.4,30.7]	[19.2,28.5]	[17.6,26.1]	[16.6,24.5]	[10.3,16.9]
35	Mayrouba	[-1.4,11.2]	[-1.7,10.2]	[16.1,16.1]	[13.5,20.3]	[10.9,25.1]	[16.5,26.5]	[22.3,28.6]	[10.5,32.2]	[7.6,27.6]	[6.4,24.3]	[0.7,23.6]	[-0.9,16.0]
36	Mchaytiyeh	[1.2,10.6]	[1.1,10.5]	[3.6,15.9]	[6.6,18.9]	[9.4,23.8]	[12.6,27.0]	[15.4,31.9]	[18.1,33.4]	[13.8,28.1]	[10.7,23.6]	[7.4,20.6]	[2.2,12.8]
37	Mymis	[6.5,14.6]	[6.3,15.2]	[8.4,19.2]	[10.2,22.2]	[13.4,26.5]	[16.5,29.4]	[17.1,31.1]	[20.1,33.7]	[17.4,30.8]	[15.6,27.6]	[13.1,25.3]	[7.7,17.2]
38	Terbol	[2.6,13.8]	[2.7,14.6]	[5.2,19.4]	[6.9,23.0]	[9.5,27.4]	[12.6,30.5]	[14.9,34.5]	[17.1,36.7]	[13.9,32.0]	[11.3,27.0]	[5.2,25.1]	[2.0,17.0]
39	Kfarhata	[9.6,18.3]	[9.1,18.6]	[10.7,21.4]	[11.7,22.4]	[14.7,25.8]	[18.3,28.5]	[19.8,29.7]	[21.6,32.2]	[19.7,30.3]	[17.3,28.3]	[13.8,27.2]	[11.3,21.6]
40	Kferden	[3.0,13.2]	[3.2,13.8]	[5.4,19.0]	[6.9,22.6]	[12.3,26.9]	[15.3,29.9]	[18.4,33.7]	[20.0,36.9]	[14.9,32.0]	[12.6,27.0]	[7.9,23.5]	[3.1,14.8]
41	Kheyem	[7.8,16.5]	[7.1,17.1]	[8.5,21.0]	[9.1,23.9]	[12.6,27.7]	[15.8,30.5]	[17.9,31.8]	[20.2,34.6]	[17.3,31.8]	[15.2,29.4]	[13.0,26.4]	[8.9,18.7]
42	Aabdeh	[11.6,20.3]	[11.1,20.6]	[12.7,23.4]	[13.7,24.4]	[16.7,27.8]	[20.3,30.5]	[21.8,31.7]	[23.6,34.2]	[21.7,32.3]	[19.3,30.3]	[15.8,29.2]	[13.3,23.6]

Annexe B

ACP pour les données de type intervalle

Cet annexe contient une explication de la méthode ACP utilisée, appelée méthodes des centres, pour projeter les données intervalles sur un espace bidimensionnel (Cazes *et al.*, 1997).

Soit Ω un ensemble de n observations notées par $\boldsymbol{\mathcal{R}}_i,\ i \in \{1,\ldots,n\}$, décrits par p variables de type intervalle tel que $\boldsymbol{\mathcal{R}}_i = \left([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p]\right)^T$, on propose de faire pour l'ensemble Ω, une Analyse en Composantes Principales, dans le but de projeter les observations sur un espace bidimensionnel. Chaque observation $\boldsymbol{\mathcal{R}}_i$ de dimension p sera alors projetée sur $\boldsymbol{\mathcal{Q}}_i$ de dimension 2, tel que $\boldsymbol{\mathcal{Q}}_i = \left([e_i^1, f_i^1], [e_i^2, f_i^2]\right)^T$.

Tout d'abord, les centres des hyperrectangles sont calculés. Ensuite, une analyse en composantes principales classique est appliquée pour ces centres :

Soit $\boldsymbol{c}_i$ le centre de l'hyper-rectangle $\boldsymbol{\mathcal{R}}_i$. Il est calculé comme suit :

$$c_i^j = \frac{a_i^j + b_i^j}{2}\ ,\ j \in \{1,\ldots,p\}$$

$\boldsymbol{c}_i$ de dimension p est projeté sur $\boldsymbol{m}_i$ de dimension 2 en utilisant l'équation suivante :

$$m_i^q = \sum_{j=1}^{p} (c_i^j - \overline{C_j}) \cdot s_j^q\ ,\ q \in \{1,2\} \tag{B.1}$$

où $\overline{C_j}$ est la moyenne des centres $c_i^j,\ i \in \{1,\ldots,n\}$. $s_j^1,\ j \in \{1,\ldots,p\}$ et $s_j^2,\ j \in \{1,\ldots,p\}$ sont les composantes des deux vecteurs propres correspondants aux deux plus grandes valeurs propres de la matrice de covariance des $c_i^j,\ i \in \{1,\ldots,n\}$.

Connaissant s_j^q , $q \in \{1,2\}$, $j \in \{1,\ldots,p\}$, les bornes e_i^q et f_i^q du vecteur $\boldsymbol{\mathcal{Q}}_i$ sont calculées comme suit :

$$e_i^q = \sum_{j,s_j^q<0} (b_i^j - \overline{C_j}) \cdot s_j^q + \sum_{j,s_j^q>0} (a_i^j - \overline{C_j}) \cdot s_j^q \qquad \text{(B.2)}$$
$$i \in \{1,\ldots,n\} \ , \ q \in \{1,2\}$$

$$f_i^q = \sum_{j,s_j^q<0} (a_i^j - \overline{C_j}) \cdot s_j^q + \sum_{j,s_j^q>0} (b_i^j - \overline{C_j}) \cdot s_j^q \qquad \text{(B.3)}$$
$$i \in \{1,\ldots,n\} \ , \ q \in \{1,2\}$$

D'autres méthodes d'Analyse en Composantes principales pour les données de type intervalle sont également proposées par Douzal-Chouakria *et al.* (2011) et Billard et Le-Rademacher (2012).

Annexe C

Minimiser une somme d'écarts absolus pondérés

Soit à minimiser la fonction suivante :

$$F(w) = \sum_{i=1}^{n} m_i \big| x_i - w \big| \tag{C.1}$$

où $x_i, w \in I\!R$ et m_i le poids associé au point x_i.

Les dérivées à gauche et à droite de $F(w)$ sont données par :

$$F'_-(w) = \sum_{i:x_i>w} m_i - \sum_{i:x_i\leqslant w} m_i \tag{C.2}$$

$$F'_+(w) = \sum_{i:x_i\geqslant w} m_i - \sum_{i:x_i<w} m_i \tag{C.3}$$

Du fait que F est convexe et linéaire par morceaux, une condition nécessaire et suffisante pour que w minimise F est :
$F'_-(w) \leqslant 0$ et $F'_+(w) \geqslant 0$. Ceci entraîne que :

$$\sum_{i:x_i<w} m_i \leqslant \sum_{i:x_i=w} m_i + \sum_{i:x_i>w} m_i \tag{C.4}$$

$$\sum_{i:x_i>w} m_i \leqslant \sum_{i:x_i=w} m_i + \sum_{i:x_i<w} m_i \tag{C.5}$$

Donc, w qui minimise F est le point x_i tel que la somme des poids des points qui le précèdent est inférieure ou égale à la somme de son poids et ceux des points qui le

succèdent, et la somme de son poids et ceux des points qui le précèdent est supérieure ou égale à la somme des poids des points qui le succèdent (en supposant que les $x_i, i \in \{1, \ldots, n\}$ sont triés par ordre croissant). w est appelé alors la **médiane pondérée** des $x_i, i \in \{1, \ldots, n\}$ (Gurwitz, 1990).

Annexe D

Minimiser une somme d'écarts au carré pondérés

Soit à minimiser la fonction suivante :

$$F(\boldsymbol{w}) = \sum_{i=1}^{n} m_i \big(x_i - w\big)^2 \tag{D.1}$$

où $x_i, w \in \mathbb{R}$ et m_i le poids associé au point x_i. $F(w)$ atteint son minimum quand $F'(w) = 0$, ceci entraîne que :

$$w = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i} \tag{D.2}$$

w est la **moyenne pondérée** des $x_i,\ i \in \{1, \ldots, n\}$.

Références bibliographiques

J. Gary Augustson et Jack Minker : An analysis of some graph theoretical cluster techniques. *J. ACM*, 17(4):571–588, octobre 1970.

F. Baçao, V. Lobo et M. Painho : Self-Organizing Maps as substitutes for K-means clustering. *In Computational Science (ICCS 2005)*, volume 3516 de *Lecture Notes in Computer Science*, pages 9–28. Springer Berlin-Heidelberg, 2005.

H-U. Bauer et K.R. Pawelzik : Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Transactions on Neural Networks*, 3 (4):570–579, 1992.

A. Ben-Hur, D. Horn, H.T. Siegelmann et V. Vapnik : Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

J.C. Bezdek : *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981. ISBN 0306406713.

J.C. Bezdek et N. R. Pal : Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics*, 28(3):301–315, 1998.

L. Billard et E. Diday : Regression analysis for interval-valued data. *In* H. Kiers, P. Groenen, J.-P. Rasson et Scader M., éditeurs : *Data Analysis, Classification, and Related Methods, Proc. IFCS2000, Namur, Belgium*. Springer-Verlag, July 11-14 2000.

L. Billard et J. Le-Rademacher : Principal component analysis for interval data. *Wiley Interdisciplinary Reviews : Computational Statistics*, 4(6):535–540, 2012. ISSN 1939-0068. URL `http://dx.doi.org/10.1002/wics.1231`.

H.H. Bock : Clustering methods and Kohonen maps for symbolic data. *J Jpn Soc Comput Stat*, 15(2):217–229, 2003.

H.H. Bock : Probabilistic modeling for symbolic data. *In* P. Brito, éditeur : *COMPSTAT 2008*, pages 55–65. Physica-Verlag HD, 2008a.

H.H. Bock : *Visualizing Symbolic Data by Kohonen Maps*, pages 205–234. John Wiley & Sons, Ltd, 2008b.

G. Cabanes, Y. Bennani, R. Destenay et A. Hardy : A new topological clustering algorithm for interval data. *Pattern Recognition*, 46(11):3030–3039, 2013.

I. Cadez, P. Smyth, G.J. McLachlan et C.E. McLaren : Maximum likelihood estimation of mixture densities for binned and truncated multivariate data. *Machine Learning*, 47:7–34, 2002.

T. Calinski et J Harabasz : A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

P. Cazes, A. Chouakria, E. Diday et Y. Schektman : Extension de l'analyse en composantes principales à des données de type intervalle. *Revue de Statistique Appliquée*, 14(3):5–24, 1997.

M. Chavent : *Analyse des données smboliques. Une méthode divisive de classification*. Thèse de doctorat, Université de Paris-IX Dauphine, 1997.

M. Chavent : An Hausdorff distance between hyper-rectangles for clustering interval data. *In* D. Banks, L. House, F. R. McMorris, F. R. Arabie et W. Gaul, éditeurs : *Classification, Clustering AND Data Mining Applications*, pages 333–340. Springer, 2004.

M. Chavent et Y. Lechevallier : Dynamical clustering of interval data : Optimization of an adequacy criterion based on Hausdorff distance. *In* K. Jajuga, A. Sokolowski et H. H. Bock, éditeurs : *Classification, Clustering and Data Analysis*, pages 53–60, Berlin, Germany, 2002. Springer. Also in the proceedings of IFCS2002, Poland.

M. Chavent et J. Saracco : On central tendency and dispersion measures for intervals and hypercubes. *Communications in Statistics - Theory and Methods*, 37(9):1471–1482, 2008.

Y. Cheng : Convergence and ordering of Kohonen's batch map. *Neural Computation*, 9(8):1667–1676, 1997.

D. Chi : Self-Organizing Map-based color image segmentation with k-means clustering and saliency map. *ISRN Signal Processing*, 2011(Article ID 393891): 18 pages, 2011.

A. Chouakria : *Extension des méthodes d'analyse factorielle à des données de type intervalle*. Thèse de doctorat, Université Paris 9 Dauphine, 1998.

D.L. Davies et D.W. Bouldin : A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

F. A. T De Carvalho : Fuzzy c-means clustering methods for symbolic interval data. *Pattern Recognition Letters*, 28(4):423–437, 2007.

F. A. T. De Carvalho, P. Brito et H. H. Bock : Dynamic clustering for interval data based on L_2 distance. *Comput Stat*, 21(2):231–250, June 2006.

F.A.T. De Carvalho : Proximity Coefficients between Boolean symbolic objects. *In* E. Diday, Y. Lechevallier, M. Schader, P. Bertrand et B. Burtschy, éditeurs : *New Approaches in Classification and Data Analysis*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 387–394. Springer Berlin Heidelberg, 1994. ISBN 978-3-540-58425-4.

F.A.T. De Carvalho : Histograms in symbolic data analysis. *Annals of Operations Research*, 55(2):299–322, 1995.

F.A.T. De Carvalho, F.M. De Melo et P. Bertrand : Batch Self-Organizing Maps based on city-block distances for interval variables. *In Proceedings of the 20th International Conference on Computational Statistics (COMPSTAT 2012)*, pages 155–166, 2012.

F.A.T. De Carvalho et R. M. C. R. De Souza : Unsupervised pattern recognition models for mixed feature-type symbolic data. *Pattern Recognition Letters*, 31:430–443, 2010.

F.A.T. De Carvalho et L.D.S Pacifico : Une version batch de l'algorithme SOM pour des données de type intervalle. *In Actes des XVIIIèmes Rencontres de la Société Francophone de Classification*, pages 99–102, 2011.

R. M. C. R. De Souza et F. A. T. De Carvalho : Clustering of interval data based on city-block distances. *Pattern Recognition Letters*, 25(3):353–365, 2004.

R. M. C. R. De Souza, F. A. T. De Carvalho et D.F. Pizzato : A partitioning method for mixed feature-type symbolic data using a squared Euclidean distance. *In* C. Freksa, M. Kohlhase et K. Schill, éditeurs : *KI 2006 : Advances in Artificial Intelligence*, volume 4314 de *Lecture Notes in Computer Science*, pages 260–273. Springer Berlin Heidelberg, 2007.

R. M. C. R. De Souza, F. A. T. De Carvalho, C. P. Tenório et Y. Lechevallier : Dynamic cluster methods for interval data based on Mahalanobis distances. *In Proceedings of the 9th Conference of the International Federation of Classification Societies*, pages 351–360, Chicago, USA, July 2004. Springer-Verlag.

A. P. Dempster, N. M. Laird et D. B. Rubin : Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of the Royal Statistical Society*, B 39:1–38, 1977.

E. Diday : La méthode des nuées dynamiques. *Revue de Statistique Appliqueée*, 19(2):19–34, 1971.

E. Diday : Introduction à l'approche symbolique en analyse des Données. *In Première Journées Symbolique-Numérique*, Université Paris IX Dauphine, 1987.

E. Diday : Introduction à l'approche symbolique en analyse des données. *Revue d'Automatique, d'Informatique et de Recherche Opérationnelle*, 23(2), 1989.

T.N. Do et F. Poulet : Kernel-based algorithms and visualization for interval data mining. *In* D.A. Zighed, S. Tsumoto, Z.W. Ras et H. Hacid, éditeurs : *Mining Complex Data*, volume 165 de *Studies in Computational Intelligence*, pages 75–91. 2009.

M.A.O. Domingues, R.M.C.R. De Souza et F.J.A. Cysneiros : A robust method for linear regression of symbolic interval data. *Pattern Recognition Letters*, 31(13):1991–1996, 2010.

D. Dong et M. Xie : Color clustering and learning for image segmentation based on neural networks. *IEEE Transactions on Neural Networks*, 6(4):925–936, 2005.

A. Douzal-Chouakria, L. Billard et E. Diday : Principal component analysis for interval-valued observations. *Statistical Analysis and Data Mining*, 4(2):229–246, 2011. ISSN 1932-1872.

J. C. Dunn : A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

A. El Golli, B. Conan-Guez et F. Rossi : Self-Organizing Maps and symbolic data. *J Symb Data Anal (JSDA)*, 2(1), 2004.

Y. El-Sonbaty et M.A. Ismail : Fuzzy clustering for symbolic data. *IEEE Transactions on Fuzzy Systems*, 6(2):195–204, 1998.

H. Elghazel : *Classification et Prévision des données hétérogènes : Application aux trajectoires et séjours hospitaliers*. Thèse de doctorat, Université Claude Bernard Lyon 1, 2007a.

H. Elghazel : *Classification et Prévision des Données Hétérogènes : Application aux Trajectoires et Séjours Hospitaliers*. Thèse de doctorat, Université Claude Bernard Lyon 1, 2007b.

M. Ester, H.P. Kriegel, J. Sander et X. Xu : A density-based algorithm for discovering clusters in large spatial databases with noise. *In* E. Simoudis, J. Han et U.M. Fayyad, éditeurs : *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press, 1996.

A. Fiannaca, G. Di Fatta, R. Rizzo, A. Urso et S. Gaglio : Simulated annealing technique for fast learning of SOM networks. *Neural Computing and Applications*, pages 1–11, 2011.

E. Forgy : Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21:768–780, 1965.

E. B. Fowlkes et C. L. Mallows : A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

C. Fraley et A.E. Raftery : How many clusters ? which clustering method ? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

B. Fritzke : *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

F. Gioia et Lauro C.N. : Principal Component Analysis on Interval Data. *Computational Statistics*, 21(2):343–363, 2006. ISSN 0943-4062.

M. L. Gonçalves, M. L. A. Netto, J. A. F. Costa et J. Zullo Júnior : An unsupervised method of classifying remotely sensed images using kohonen self-Organizing Maps and agglomerative hierarchical clustering methods. *International Journal of Remote Sensing*, 29(11):3171–3207, 2008.

G. Govaert : *Classification automatique et distances adaptatives*. Thèse de doctorat, Université Paris VI, 1975.

K.C. Gowda et E. Diday : Symbolic clustering using a new dissimilarity measure. *Pattern Recognition Letters*, 24(6):567–578, 1991.

C. Gurwitz : Weighted median algorithms for L_1 approximation. *BIT*, 30 (2):301–310, 1990.

C. Hajjar et H. Hamdan : Self-Organizing Map based on city-block distance for interval-valued data. *In Complex Systems Design & Management*, pages 281–292, Paris, France, December 7-9 2011a.

C. Hajjar et H. Hamdan : Self-Organizing Map based on Hausdorff distance for interval-valued data. *In IEEE International Conference on Systems, Man, and Cybernetics*, pages 1747–1752, Anchorage, Alaska, October 9-12 2011b.

C. Hajjar et H. Hamdan : Self-Organizing Map based on L_2 distance for interval-valued data. *In IEEE International Symposium on Applied Computational Intelligence and Informatics*, pages 317–322, Timisoara, Romania, May 19-21 2011c.

C. Hajjar et H. Hamdan : *Applied Computational Intelligence in Engineering and Information Technology*, chapitre Clustering of interval data using Self-Organizing Maps - Application to meteorological data, pages 135–146. Topics in Intelligent Engineering and Informatics. Springer-Verlag, 2012a.

C. Hajjar et H. Hamdan : Kohonen neural networks for interval-valued data clustering. *International Journal of Advanced Computer Science*, 2(11):412–419, 2012b.

C. Hajjar et H. Hamdan : Self-Organizing Maps for mixed feature-type symbolic data. *In IEEE International Symposium on Signal Processing and Information Technology*, pages 135–140, Ho Chi Minh City, Vietnam, December 12-15 2012c.

C. Hajjar et H. Hamdan : Interval data clustering using Self-Organizing Maps based on adaptive Mahalanobis distances. *In International Joint Conference on Neural Networks*, pages 1044–1049, Dallas, USA, August 4-9 2013a.

C. Hajjar et H. Hamdan : Interval data clustering using Self-Organizing Maps based on adaptive Mahalanobis distances. *Neural Networks*, 46:124–132, 2013b.

C. Hajjar et H. Hamdan : Cartes auto-organisatrices pour la classification des données de type intervalle en se basant sur la distance city-block. *Revue des Nouvelles Technologies de l'Information*, 2014. Accepté.

H. Hamdan : *Développement de méthodes de classification pour le contrôle par émission acoustique d'appareils à pression.* Thèse de doctorat, Université de technologie de Compiègne, 2005.

H. Hamdan : Mixture model clustering of binned uncertain data : the classification approach. *In IEEE International Conference on Information & Communication Technologies : from Theory to Applications*, pages 1645–1650, Damascus, Syria, April 24-28 2006.

H. Hamdan et G. Govaert : Classification de données de type intervalle via l'algorithme EM. *In XXXV^{èmes} Journées de Statistique, SFdS*, pages 549–552, Lyon, France, 2-6 juin 2003a.

H. Hamdan et G. Govaert : Modélisation et classification de données imprécises. *In CIMNA'03, premier congrès international sur les modélisations numériques appliquées*, pages 16–19, Beyrouth, Liban, 14-15 novembre 2003b.

H. Hamdan et G. Govaert : CEM algorithm for imprecise data. Application to flaw diagnosis using acoustic emission. *In IEEE International Conference on Systems, Man and Cybernetics*, pages 4774–4779, The Hague, The Netherlands, October 10-13 2004a.

H. Hamdan et G. Govaert : The fitting of binned data clustering to imprecise data. *In IEEE International Conference on Information & Communication Technologies : from Theory to Applications*, pages 1–6, Damascus, Syria, April 19-23 2004b.

H. Hamdan et G. Govaert : Int-EM-CEM algorithm for imprecise data. Comparison with the CEM algorithm using Monte Carlo Simulations. *In IEEE International Conference on Cybernetics and Intelligent Systems*, pages 410–415, Singapore, December 1-3 2004c.

H. Hamdan et G. Govaert : Mixture model clustering of uncertain data. *In IEEE International Conference on Fuzzy Systems*, pages 879–884, Reno, Nevada, USA, 22-25 May 2005.

H. Hamdan et C. Hajjar : A neural networks approach to interval-valued data clustering. Application to Lebanese meteorological stations data. *In IEEE Workshop on Signal Processing Systems*, pages 373–378, Beirut, Lebanon, October 4-7 2011.

H. Hamdan et J. Wu : EM algorithm of spherical models for binned data. *In*

IEEE International Symposium on Signal Processing and Information Technology, pages 99–105, Bilbao, Spain, December 14-17 2011.

H. Hamdan et J. Wu : Bin-EM-CEM algorithms of spherical parsimonious Gaussian mixture models for binned data clustering. *In IEEE International Conference on Intelligent Engineering Systems*, pages 187–192, Costa Rica, June 19-21 2013a.

H. Hamdan et J. Wu : Model selection with BIC and ICL criteria for binned data clustering by bin-EM-CEM algorithms. *In IEEE International Conference on Systems, Man, and Cybernetics*, pages 3133–3138, Manchester, United Kingdom, October 13-16 2013b.

T. Heskes : *Kohonen Maps*, pages 303–315. Elsevier, Amesterdam, 1999a.

Tom Heskes : Energy Functions for Self-Organizing Maps. 1999b.

Zhexue Huang : Clustering large data sets with mixed numeric and categorical values. *In In The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 21–34, 1997a.

Zhexue Huang : A fast clustering algorithm to cluster very large categorical data sets in data mining. *In In Research Issues on Data Mining and Knowledge Discovery*, pages 1–8, 1997b.

L. Hubert et P. Arabie : Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

M. Ichino et H. Yaguchi : Generalized minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 24 (4):698–708, 1994.

S. Johnson : Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

L. Kaufman et P. Rousseeuw : *Clustering by Means of Medoids*. Reports of the Faculty of Mathematics and Informatics. Delft University of Technology. 1987.

J. Kim et L. Billard : Dissimilarity measures and divisive clustering for symbolic multimodal-valued data. *Computational Statistics & Data Analysis*, 56(9):2795–2808, 2012.

K. Kiviluoto : Topology preservation in Self-Organizing Maps. *In International Conference on Neural Networks*, pages 294–299, 1996.

T. Kohonen : *Self organization and associative memory, Second Edition*. Springer-Verlag, 1984.

T. Kohonen : *Self-Organizing Maps, Third Edition*. Springer, 2001.

P. Koikkalainen : Fast Deterministic Self-Organizing Maps. *In Proceedings of the International Conference on Artificial Neural Networks (ICANN95)*, volume II, pages 63–68, 1995a.

P. Koikkalainen : Fast deterministic self-Organizing Maps. *In Proceedings of the International Conference on Artificial Neural Networks (ICANN'95)*, volume II, pages 63–68, 1995b.

J. H. Lee, D. H. Kim et C. W. Chung : Multi-dimensional selectivity estimation using compressed histogram information. *In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 205–214. ACM Press, 1999. ISBN 1-58113-084-8.

J. B. MacQueen : Some Methods for Classification and Analysis of Multivariate Observations. *In Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

S. Makosso K. : *Analyse en Composantes Principales de variables symboliques de type histogramme*. Thèse de doctorat, Paris Dauphine-Ceremade, 2010.

T. Martinetz et K. Schulten : A "Neural-Gas" network learns topologies. *In* K. Kohonen, T.and Mäkisara, O. Simula et J. Kangas, éditeurs : *Artificial Neural Networks*, pages 397–402. Elsevier, 1991.

Y. Matias, J.S. Vitter et M. Wang : Wavelet-based histograms for selectivity estimation. *In Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 448–459. ACM Press, 1998. ISBN 0-89791-995-5.

G. J. McLachlan et P. N. Jones : Fitting mixture models to grouped and truncated data via the EM algorithm. *Biometrics*, 44(2):571–578, 1988.

D. Merkl : Document classification with Self-Organizing Maps. *In* E. Oja et S. Kaski, éditeurs : *Kohonen Maps*, pages 183–192, Amesterdam, The Netherlands, 1999. Elsevier.

R. Miikkulainen : Script Recognition with Hierarchical Feature Maps. *Connection Science*, 2:83–101, 1990.

J. Moreira et L. D. F. Costa : Neural-based color image segmentation and classification using Self-Organizing Maps. *In Anais do IX Simpó,sio Brasileiro de Computacõ, Grá,fica e Processamento de Imagens (SIBGRAPI96)*, pages 47–54, 1996.

V. Poosala : *Histogram-based estimation techniques in database systems*. Thèse de doctorat, University of Wisconsin at Madison, 1997.

V.V. Raghavan et K. Birchard : A clustering strategy based on a formalism of the reproductive process in natural systems. *In Proceedings of the 2nd annual International ACM SIGIR Conference on Information Storage and Retrieval : information implications into the eighties*, SIGIR79, pages 10–22, New York, NY, USA, 1979.

H. Ralambondrainy : A conceptual version of the K-means algorithm. *Pattern Recognition Letters*, pages 1147–1157, 1995.

D.M. Ristić, M. Pavlović et I. Reljin : Image segmentation method based on Self-Organizing Maps and K-Means algorithm. *In Proceedings of the 9th Symposium on Neural Network Applications in Electrical Engineering (NEUREL08)*, pages 27–30, Belgrade, Serbia, September 2008.

H. Ritter, T. Martinetz et S. Schulten : *Neural Computation and Self-Organizing Maps : An Introduction*. Addison-Wesley, Reading, Massachussets, 1992.

H. Ritter et K. Schulten : On the stationary state of Kohonen's self-organizing sensory mapping. *Biological Cybernetics*, 54:99–106, 1986.

H. Ritter et S. Schulten : Kohonen's Self-Organizing Maps : exploring their computational capabilities. *In Proceedings of IEEE International Conference on Neural Networks*, volume 1, pages 109–116, July 1988.

F. Rossi et B. Conan-Guez : Multi-Layer Perceptron on interval data. *In* K. Jajuga, A. Sokolowski et H. H. Bock, éditeurs : *Classification, Clustering, and Data Analysis*, pages 427–436. Springer, Berlin, Germany, 2002.

P. J. Rousseeuw : Silhouettes : A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(0):53–65, 1987.

A. Samé : *Modèles de mélange et classification de données acoustiques en temps réel*. Thèse de doctorat, Université de Technologie de Compiègne, 2004.

A. Samé, C. Ambroise et G. Govaert : A classification EM algorithm for binned data. *Computational Statistics and Data Analysis*, 51(2):466–480, 2006.

R. Sibson : Slink : An optimally efficient algorithm for the single-link cluster method. *Computer Journal*, 16(1):30–34, 1973.

M.C. Su et Chang H.T. : Fast Self-Organizing Feature Map Algorithm. *IEEE Transactions on Neural Networks*, 11(3):721–733, 2000.

R. Verde : Clustering Methods in Symbolic Data Analysis. *In* D. Banks, F. McMorris, P. Arabie et W. Gaul, éditeurs : *Classification, Clustering, and Data Mining Applications*, Studies in Classification, Data Analysis, and Knowledge Organisation, pages 299–317. 2004.

Rosanna Verde et Antonio Irpino : Dynamic clustering of histogram data : using the right metric. *In* P. Brito, G. Cucumel, P. Bertrand et F.A.T. De Carvalho, éditeurs : *Selected Contributions in Data Analysis and Classification*, pages 123–134. Springer-Berlin Heidelberg, 2007.

J. Vesanto et E. Alhoniemi : Clustering of the Self-Organizing Map. *IEEE Transactions on Neural Networks*, 11(3):586–600, May 2000.

W. Wang, J. Yang et R.R. Muntz : STING : A statistical information grid approach to spatial data mining. *In* M. Jarke, M.J. Michael J. Carey, K.R. Dittrich, F.H. Lochovsky, P. Loucopoulos et M.A. Jeusfeld, éditeurs : *VLDB97, Proceedings of 23rd International Conference on Very Large Databases, August 25-29, 1997, Athens, Greece*, pages 186–195. Kaufmann, M., 1997.

J. Ward : Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

J. Wu et H. Hamdan : Parsimonious Gaussian mixture models of diagonal family for binned data clustering : mixture approach. *In IEEE International Symposium on Computational Intelligence and Informatics*, pages 385–390, Budapest, Hungary, November 21-22 2011.

J. Wu et H. Hamdan : Parsimonious Gaussian mixture models of general family for binned data clustering : mixture approach. *In IEEE International Symposium on Applied Machine Intelligence and Informatics*, pages 283–288, Herl'any, Slovakia, January 26-28 2012.

J. Wu et H. Hamdan : Bin-EM-CEM algorithms of general parsimonious Gaussian mixture models for binned data clustering. *In IEEE International Conference on Computational Cybernetics*, pages 17–22, Tihany, Hungary, July 8-10 2013a.

J. Wu et H. Hamdan : Model choice for binned-EM algorithms of fourteen parsimonious Gaussian mixture models by BIC and ICL criteria. *In IEEE International Conference on System Science and Engineering*, pages 351–356, Budapest, Hungary, July 4-6 2013b.

M.S. Yang, W.L. Hung et D.H. Chen : Self-Organizing Map for symbolic data. *Fuzzy Sets and Systems*, 203(0):49–73, 2012.

M.S. Yang, P.Y. Hwang et D.H. Chen : Fuzzy clustering algorithms for mixed feature variables. *Fuzzy Sets and Systems*, 141(2):301–317, 2004.

C.T. Zahn : Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1):68–86, 1971.

Y. Zhao et G. Karypis : Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.

S. Zrehen et F. Blayo : A geometric organization measure for Kohonen maps. *In Proceedings of NeuroNîmes*, pages 611–621, 1992.